MATLAB 中文手册

VISIT…

LANZAROTE Caliente.COM

第 1 章　MATLAB 6.5 环境

1.1 MATLAB 简介

- MATLAB(Matrix Laborator)是 MathWorks 公司开发科学与工程计算软件；
- 广泛应用于自动控制、数学运算、信号分析、计算机技术、图像信号处理、财务分析、航天工业、汽车工业、生物医学工程、语音处理和雷达工程等行业；
- 国内外高校和研究部门科学研究的重要工具；
- MATLIB 已成为数学计算工具方面事实上的标准，MATLIB 6.5 是最新版本。

1.1.1 MATLAB 工具箱

- MATLAB 由基本部分和功能各异的工具箱组成。

基本部分是 MATLAB 的核心，工具箱是扩展部分。

- 工具箱是用 MATLAB 的基本语句编成的各种子程序集，用于解决某一方面的专门问题或实现某一类的新算法。
- MATLAB 有以下主要的工具箱：

▪ 控制系统工具箱(Control System Toolbox)
▪ 系统辨识工具箱(System Identification Toolbox)
▪ 信号处理工具箱(Signal Processing Toolbox)
▪ 神经网络工具箱(Neural Network Toolbox)
▪ 模糊逻辑控制工具箱(Fuzzy Logic Toolbox)
▪ 小波工具箱(Wavelet Toolbox)
▪ 模型预测控制工具箱(Model Predictive Control Toolbox)
▪ 通信工具箱(Communication Toolbox)
▪ 图像处理工具箱(Image Processing Toolbox)
▪ 频域系统辨识工具箱(Frequency System Identification Toolbox)
▪ 优化工具箱(Optimization Toolbox)
▪ 偏微分方程工具箱(Partial Differential Equation Toolbox)
▪ 财政金融工具箱(Financial Toolbox)
▪ 统计工具箱(Statistics Toolbox)

1.1.2 MATLAB 功能和特点

1．功能强大

(1) 运算功能强大

- MATLAB 的数值运算要素不是单个数据，而是矩阵，每个元素都可看作复数，运算包括加、减、乘、除、函数运算等；

- 通过 MATLAB 的符号工具箱，可以解决在数学、应用科学和工程计算领域中常常遇到的符号计算问题。

(2) 功能丰富的工具箱

大量针对各专业应用的工具箱的提供，使 MATLAB 适用于不同领域。

(3) 文字处理功能强大

MATLAB 的 Notebook 为用户提供了强大的文字处理功能，允许用户从 Word 访问 MATLAB 的数值计算和可视化结果。

2．人机界面友好，编程效率高

- 语言规则与笔算式相似，命令表达方式与标准的数学表达式非常相近。
- 解释方式工作的，键入算式无需编译立即得出结果，若有错误也立即做出反应，便于编程者立即改正。

3．强大而智能化的作图功能

- 工程计算的结果可视化，使原始数据的关系更加清晰明了；
- 多种坐标系；
- 能绘制三维坐标中的曲线和曲面。

4．可扩展性强

包括基本部分和工具箱两大部分，具有良好的可扩展性，工具箱可以任意增减。

5. Simulink 动态仿真功能

MATLAB 的 Simulink 提供了动态仿真的功能，用户通过绘制框图来模拟一个线性、非线性、连续或离散的系统，通过 Simulink 能够仿真并分析该系统。

1.2　MATLAB 6.5 环境设置

MATLAB6.5 版的界面更加方便，运行界面称为 MATLAB 操作界面(MATLAB Desktop)，默认的操作界面如图 1.1 所示。

MATLAB 的操作界面是一个高度集成的工作界面，它的通用操作界面包括九个常用的窗口，另外，MATLAB6.5 版还增加了“Start”开始按钮。

1.2.1 菜单栏

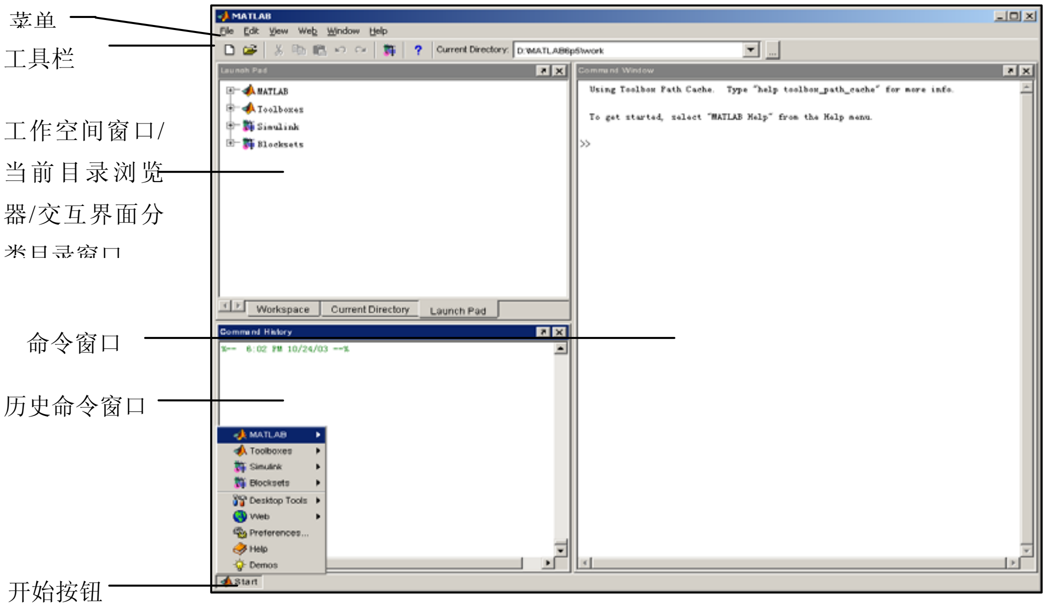


图 1.1 MATLAB 6.5 版的默认界面

MATLAB 操作界面菜单提供了“File”、“Edit”、“View”、“Web”、“Window”和“Help”菜单。

1．File 菜单

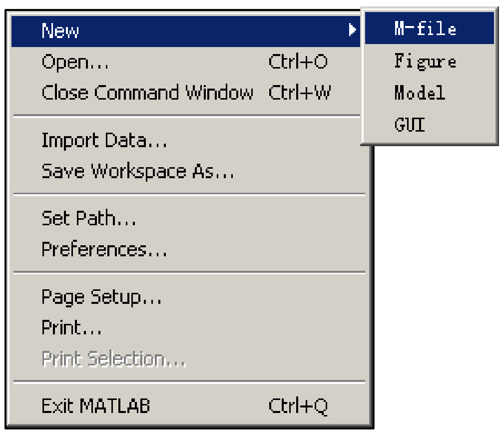


图 1.2 File 菜单

表 1.1 File 菜单功能表

下拉菜单		功能
New	M-file	新建一个 M 文件，打开 M 文件编辑/调试器
	Figure	新建一个图形窗口
	Model	新建一个仿真模型
	GUI	新建一个图形用户设计界面(GUI)
Open...		打开已有文件
Close Command History		关闭历史命令窗口
Import Data...		导入其他文件的数据
Save Workspace as...		使用二进制的 MAT 文件保存工作空间的内容
Page Setup...		页面设置
Set Path...		设置搜索路径等

Preferences...	设置 MATLAB 工作环境外观和操作的相关属性等参数
Print...	打印
Print Selection...	打印所选择区域
Exit MATLAB	退出 MATLAB

2．Edit 菜单

- Edit 菜单如图 1.3 所示，Edit 菜单的各菜单项与 Windows 的 Edit 菜单相似；
- “Paste Special”有点特殊，可以用来打开数据输入向导对话框“Import Wizard”，将剪贴板的数据输入到 MATLAB 工作空间中。

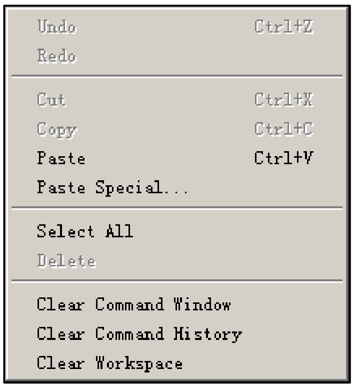


图 1.3 Edit 菜单

3．View 菜单

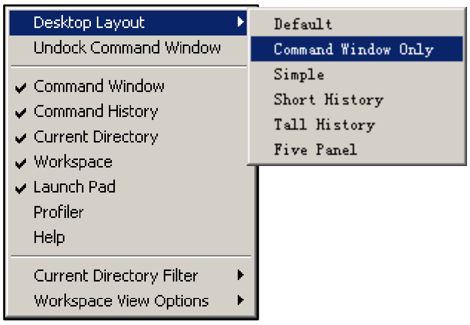


图 1.4 View 菜单

表 1.2 View 菜单功能表

下拉菜单	功能
Desktop Layout	界面布局(可选择各种布局方式)
Undock Command Window	与命令窗口分离
Command Window	打开命令窗口
Command History	打开历史命令窗口
Current Directory	打开当前目录窗口
Workspace	打开工作空间窗口
Launch Pad	打开交互界面分类目录窗口

Profiler	打开程序性能剖析窗口
Help	打开帮助窗口

4. Web 菜单

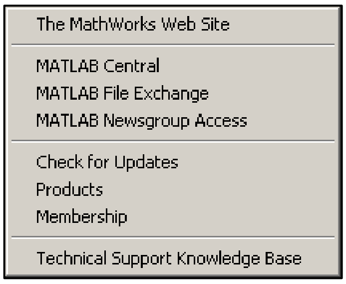


图 1.5 Web 菜单

表 1.3 Web 菜单功能表

下拉菜单	功能
The MathWorks Web Site	连接到 MathWorks 公司的主页
MATLAB Central	连接到 MATLAB Central
MATLAB File Exchange	连接到 MATLAB File Exchange
MATLAB Newsgroup Access	连接到 MATLAB Newsgroup Access
Check for Updates	通过网站检查版本更新
Products	连接到产品介绍页面
Membership	连接到介绍 MathWorks 公司的会员制度
Technical Support Knowledge Base	连接到 MathWorks 公司的技术支持网页

5. Windows 菜单

Windows 菜单提供了在已打开的各窗口之间切换的功能。

6. Help 菜单

Help 菜单提供了进入各类帮助系统的方法。

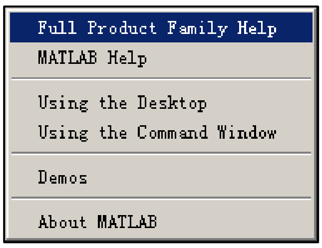


图 1.6 Help 菜单

7. 开始菜单

- 上半部分是交互界面窗口的列表；
- 下半部分是常用的子菜单项，包括：Desktop Tools、Web、Preferences、Help 和 Demos。

1.2.2 工具栏

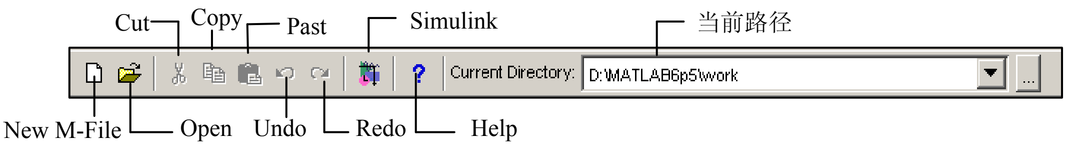


图 1.8 工具栏

1.2.3 通用操作界面窗口

1. 命令窗口(Command Window)

在命令窗口中可键入各种 MATLAB 的命令、函数和表达式，并显示除图形外的所有运算结果。

- 命令窗口单独显示：如果选择菜单“View”→“Undock Command Window ”；
- 单独的命令窗口返回 MATLAB 界面：选择命令窗口的菜单“View”→“Dock Command Window”命令。

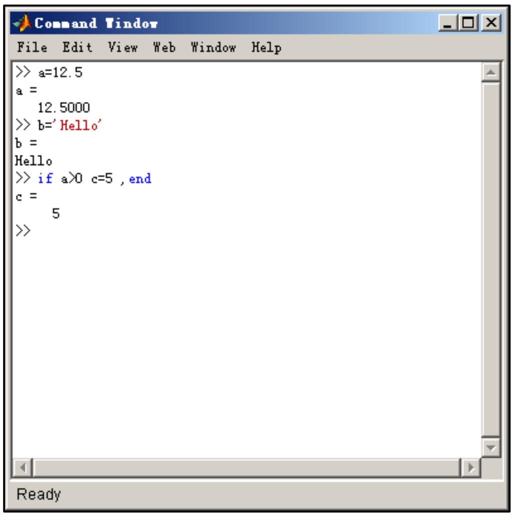


图 1.9 单独的命令窗口

(1) 命令行的显示方式

- 命令窗口中的每个命令行前会出现提示符“>>”。
- 命令窗口内显示的字符和数值采用不同的颜色，在默认情况下，输入的命令、表达式以及计算结果等采用黑色字体；
- 字符串采用赭红色；“if”、“for”等关键词采用蓝色。

【例 1.1】在命令窗口中输入不同的数值和语句，并查看其显示方式。

```
>> a=12.7
```

```
a =
   12.7000
```

```
>> b='Hello'

b =
Hello

>> if a>0 c=5 ,end

c =
     5
```

(2) 命令窗口中命令行的编辑

MATLAB 命令窗口不仅可以对输入的命令进行编辑和运行，而且可以对已输入的命令进行回调、编辑和重运行。常用操作键如表 1.4 所示。

表 1.4　命令窗口中行编辑的常用操作键

键名	作用	键名	作用
↑	向前调回已输入过的命令行	Home	使光标移到当前行的开头
↓	向后调回已输入过的命令行	End	使光标移到当前行的末尾
←	在当前行中左移光标	Delete	删去光标右边的字符
→	在当前行中右移光标	Backspace	删去光标左边的字符
PageUp	向前翻阅当前窗口中的内容	Esc	清除当前行的全部内容
Page Down	向后翻阅当前窗口中的内容	CTRL+C	中断 MATLAB 命令的运行

(3) 命令窗口中的标点符号

表 1.5　MATLAB 常用标点符号的功能

名称	符号	功能
空格		用于输入变量之间的分隔符以及数组行元素之间的分隔符。
逗号	,	用于要显示计算结果的命令之间的分隔符；用于输入变量之间的分隔符；用于数组行元素之间的分隔符。
点号	.	用于数值中的小数点。
分号	;	用于不显示计算结果命令行的结尾；用于不显示计算结果命令之间的分隔符；用于数组元素行之间的分隔符。
冒号	:	用于生成一维数值数组，表示一维数组的全部元素或多维数组的某一维的全部元素。
百分号	%	用于注释的前面，在它后面的命令不需要执行。
单引号	‘ ’	用于括住字符串。
圆括号	( )	用于引用数组元素；用于函数输入变量列表；用于确定算术运算的先后次序。
方括号	[ ]	用于构成向量和矩阵；用于函数输出列表。
花括号	{ }	用于构成元胞数组。
下划线	-	用于一个变量、函数或文件名中的连字符。
续行号	…	用于把后面的行与该行连接以构成一个较长的命令。
“At”号	@	用于放在函数名前形成函数句柄；用于放在目录名前形成用户对象类目录。

注意：以上的符号一定要在英文状态下输入，因为 MATLAB 不能识别中文标点符号。

【例 1.2】在命令窗口中使用不同的标点符号。

```
>> a=12.5,b='Hello'   %逗号表示分隔命令,单引号构成字符串,点号为小数点

a =
   12.5000
b =
Hello

>>c=[1 2;3 4;5 6]          % [ ]表示构成矩阵,分号用来分隔行,空格用来分隔元素

c =
     1     2
     3     4
     5     6

>> d=a*...                %...表示续行
```

(4) 数值计算结果的显示格式及设置

- 默认显示格式为：当数值为整数，以整数显示；当数值为实数，以小数后 4 位的精度近似显示，即以“短(Short)”格式显示；如果数值的有效数字超出了这一范围，则以科学计数法显示结果。
- 显示格式设置：选择菜单“File”→“ Preferences”，则会出现参数设置对话框，如图 1.10 所示；

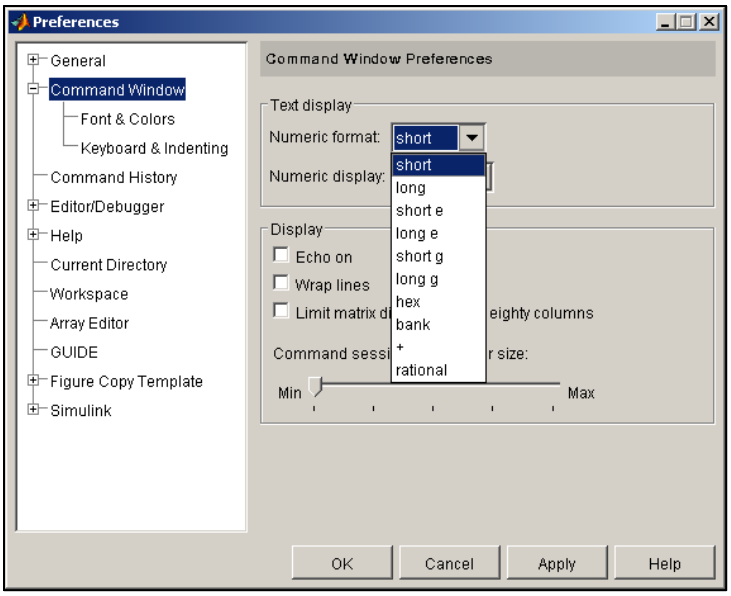


图 1.10 参数设置对话框

- 直接在命令窗口中输入 “format” 命令来进行数值显示格式的设置。

format 格式描述

表 1.6　数据显示的 Format 格式

命令格式	含义	例子
format	通常保证小数点后四位有效；大于 1000 的	314.159 显示为 314.1590

format short(默认)	实数，用 5 位有效数字的科学计数法显示	3141.59 显示为 3.1416e+003
format short e	5 位科学计数法表示	π 显示为 3.1416e+000
format short g	从 format short 和 format short e 中自动选择最佳计数方式	π 显示为 3.1416
format long	15 位数字表示	π 显示为 3.14159265358979
format long e	15 位科学计数法表示	π 显示为 3.141592653589793e+000
format long g	从 format long 和 format long e 中自动选择最佳计数方式	π 显示为 3.1415926358979
format rat	近似有理数表示	π 显示为 355/113
format hex	十六进制表示	π 显示为 400921fb54442dl8
format +	正数、负数、零分别用+、一、空格	π 显示为+
format bank	表示(金融)元、角、分	π 显示为 3.14
format compact	在显示结果之间没有空行的压缩格式	
format loose	在显示结果之间有空行的稀疏格式	

(5) 命令窗口的常用控制命令

- clc：用于清空命令窗口中的显示内容。
- more：在命令窗口中控制其后每页的显示内容行数。

2. 历史命令窗口(Command History)

表 1.7　历史指令窗口主要功能的操作方法

应用功能	操作方法
单行或多行命令的复制(Copy)	选中单行或多行命令，按鼠标右键出现快捷菜单，再选择“Copy”菜单，就可以把它复制。
单行或多行命令的运行(Evaluate Selection)	选中单行或多行命令，按鼠标右键出现快捷菜单，再选择“Evaluate Selection”菜单，就可在命令窗口中运行，并得出相应结果。 或者双击选择的命令行也可运行。
把多行命令写成 M 文件(Create M-File)	选中单行或多行命令，按鼠标右键出现快捷菜单，选择“Create M-File”菜单，就可以打开写有这些命令的 M 文件编辑/调试器窗口。

例如，复制和运行图 1.11 所示历史命令窗口中的前三行命令。

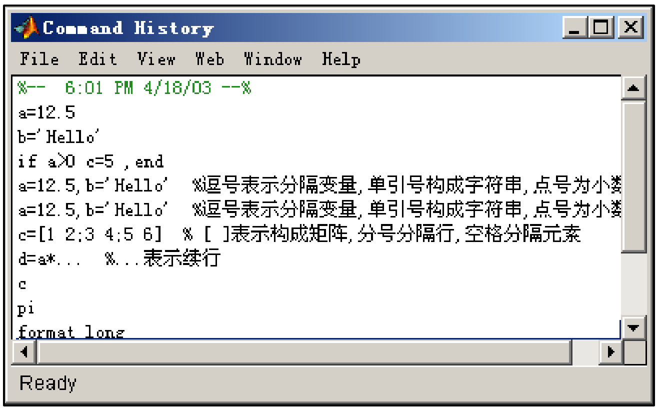


图 1.11 历史命令窗口

3. 当前目录浏览器窗口(Current Directory Browser)

(1) 当前目录的设置

如果是通过单击 Windows 桌面上的 MATLAB 图标启动，则启动后的默认当前目录是“matlab/work”；

如果 MATLAB 的启动是由单击“matlab/bin/win32”目录下的“matlab.exe”，则默认当前目录是“matlab/bin/win32”。

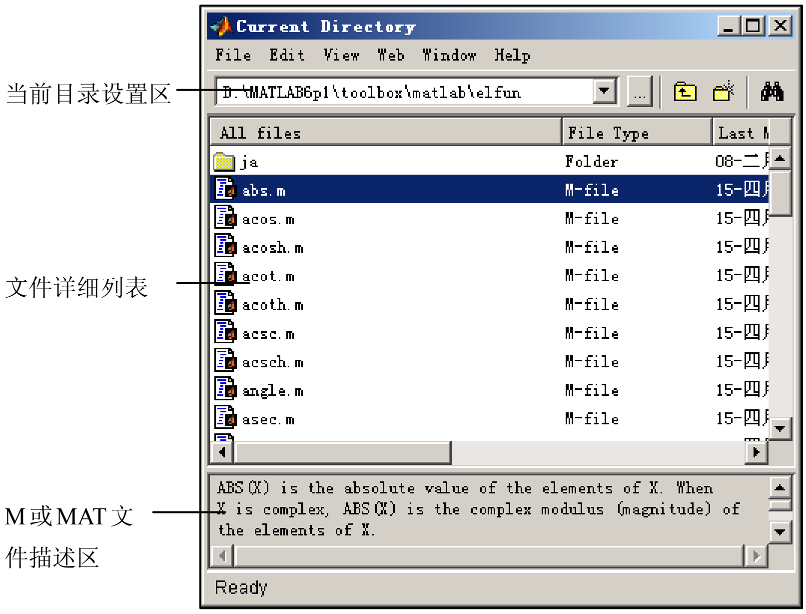


图 1.13 当前目录浏览器窗口

把用户目录设置成当前目录的方法有两种：

- 在当前目录设置区设置。在上图中或 MATLAB 界面工具栏的右边都有当前目录设置区， 可以在“设置栏”中直接填写待设置的目录名。
- 通过命令设置

cd	%显示当前目录
cd 目录	%指定当前目录
cd ..	%指定上一级目录为当前目录

(2) 文件详细列表区的使用

表 1.8 文件详细列表区的主要应用功能

功能	操作方法
运行 M 文件(Run)	选择待运行文件，按鼠标右键出现快捷菜单，选择 “Run” 菜单 运行 M 文件。
打开 M 文件(Open)	选择待运行 M 文件，按鼠标右键出现快捷菜单，选择“Open”菜单，则 M 文件出现在 M 文件编辑/调试器窗口中。 或者双击该 M 文件也可打开文件。
把 MAT 文件全部数据输入内存(Open)	选择待装入的 MAT 数据文件，按鼠标右键出现快捷菜单，选择“Open”菜单，此文件的数据就全部装入工作空间。 或者双击该 MAT 文件也可实现。
把 MAT 文件部分数据输入内存	选择待装载 MAT 数据文件，按鼠标右键出现快捷菜单，选择“Import Data”菜单，出现数据输入向导对话框“Import Wizard”，选择待装入的数据变量名，

(Import Data)	然后单击“Finish”按钮。

(3) M 或 MAT 文件描述区

显示 M 或 MAT 文件描述区：

选择菜单“File”→“preferences”，在“Preferences”对话框中点击左侧的“Current Directory”选项，在对话框的右边“Brower Display Options”中选择“Show M-file Comments and MAT-file Comments”复选框，然后单击“OK”按钮。

4. 工作空间浏览器窗口(Workspace Browser)

- 工作空间浏览器窗口用于显示所有 MATLAB 工作空间中的变量名、数据结构、类型、大小和字节数。
- 可以对变量进行观察、编辑、提取和保存。

```
a=12.5
b='Hello'
c=[1 2;3 4;5 6]
```

图 1.14 为工作空间窗口的单独窗口显示。

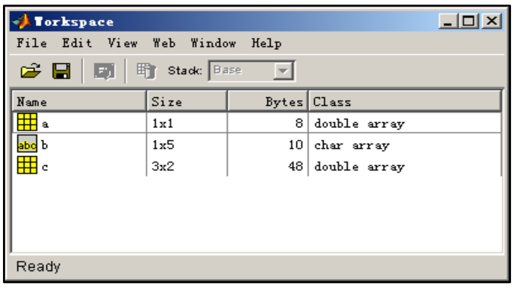


图 1.14 工作空间浏览器窗口

(1) 当前目录浏览器窗口中变量的操作

表 1.9 工作空间浏览器主要功能的操作方法

功能	操作方法
变量的字符显示	选中变量按鼠标右键出现快捷菜单，选择“Open…”菜单，则数值类、字符类变量显示在“Array Editor”数组编辑器窗口中。 或者双击该变量也可显示。
变量的图形显示	选中变量按鼠标右键出现快捷菜单，选择菜单“Graph”的下拉菜单，则系统就以该绘图命令使变量可视化显示。
全部内存变量保存为 MAT 文 件	按鼠标右键出现快捷菜单，选择“Save Workspace As…”菜单，则可把当前内存中全部变量保存为数据文件。
部分内存变量保存为 MAT 文件	选中若干变量按鼠标右键出现快捷菜单，选择“Save Selection As…”菜单，则可把所选变量保存为数据文件。
删除部分内存变量	选中一个或多个变量按鼠标右键出现快捷菜单，选择“Delete”菜单。出现“Confirm Delete”对话框，单击“Yes”按钮。

	或者选择工作空间浏览器窗口的菜单“Edit”→“Delete”。
删除全部内存变量	按鼠标右键出现快捷菜单，选择“Clear Workspace”菜单。

(2) 通过命令管理变量

- **save:** 把工作空间中的数据存放到 MAT 数据文件

save FileName 变量 1 变量 2 … 参数 %将变量保存到文件中

说明：

FileName 为 MAT 文件名；

变量 1、变量 2 可以省略，省略时则保存工作空间的所有变量；

参数为保存的方式，有-ASCII、-append 等方式。

```
>> save FileName1                    %把全部内存变量保存为 FileName1.mat 文件
>> save FileName2 a b                %把变量 a，b 保存为 FileName2.mat 文件
>> save FileName3 a b -append        %把变量 a，b 添加到 FileName3.mat 文件中
```

- **load:** 从数据文件中取出变量到工作空间

load FileName 变量 1 变量 2 …

说明：变量 1、变量 2 可以省略，省略时则装载所有变量。

例如：

```
>> load Filename1          %把 FileName1.mat 文件中的全部变量装入内存
>> load FileName2 a b      %把 FileName2.mat 文件中的 a，b 变量装入内存
```

- **who:** 查阅 MATLAB 内存变量变量名

```
>> who
Your variables are:
a   b   c
```

- **whos:** 查阅 MATLAB 内存变量变量名、大小、类型和字节数

```
>> whos
  Name        Size             Bytes   Class
  a           1x1                  8   double array
  b           1x5                 10   char array
  c           3x2                 48   double array
Grand total is 12 elements using 66 bytes
```

- **clear:** 删除工作空间中的变量

```
>> clear a
>> who
Your variables are:
b   c
```

- **exist('X') :** 查询工作空间中是否存在某个变量

i=exist('X') %查询工作空间中是否有'X'变量

说明：

- i=1 ：表示存在一个变量名为'X'的变量；
- i=2 ：表示存在一个名为'X.m'的文件；
- i=3 ：表示存在一个名为'X.mex'的文件；
- i=4 ：表示存在一个名为'X.mdl'文件；
- i=5 ：表示存在一个名为'X'的内部函数；
- i=0 ：表示不存在以上变量和文件。

5. 数组编辑器窗口(Array Editor)

打开选择数组编辑器窗口：“Open…”菜单或者双击该变量。

图 1.15 为变量“c=[1 2;3 4;5 6]”在“Array Editor”数组编辑器窗口中的显示。

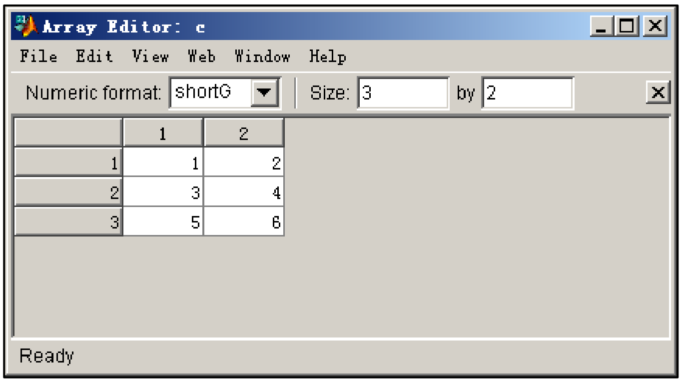


图 1.15 “Array Editor”数组编辑器窗口

- 在“Numeric format”栏中改变变量的显示类型。
- 在“Size”、“by”栏中改变数组的大小。
- 逐格修改数组中的元素值。

6. 交互界面分类目录窗口(Launch Pad)

- 双击应用条目“Import Wizard”、“Profiler”和“GUIDE”，就出现相应的界面窗口。
- 双击“Help” 条目，就打开帮助文件出现帮助导航／浏览器窗口。
- 双击“Demos”条目，就出现帮助导航／浏览器窗口的 Demos 选项卡。
- 双击“Product Page (Web)”条目，就会上网连接支持网站的相应产品页面。

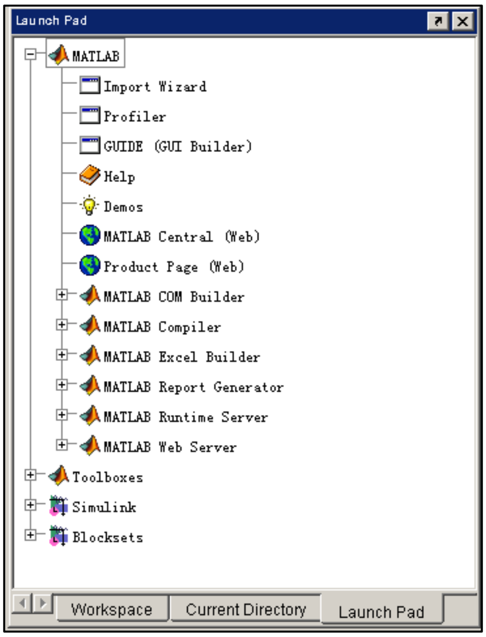


图 1.16 交互界面分类目录窗口

7. M 文件编辑／调试器窗口(Editor／Debugger)

启动 M 文件编辑／调试器窗口的方法：

- 单击 MATLAB 界面上的图标，或者单击菜单“File”→“New”→“M-file”，可打开空白的 M 文件编辑器。

- 单击 MATLAB 界面上的图标，或者单击菜单“File”→“Open”，在打开的“Open”对话框中填写所选文件名，单击“打开”按钮，就可出现相应的 M 文件编辑器。

▪ 用鼠标双击当前目录窗口中的 M 文件(扩展名为.m)，可直接打开相应文件的 M 文件编辑器。

图 1.17 显示打开了一个“Ex0101.m”文件的 M 文件编辑 / 调试器窗口：

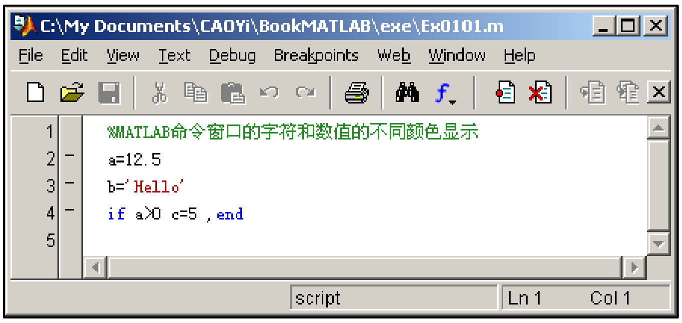


图 1.17 M 文件编辑 / 调试器窗口

8. 帮助导航 / 浏览器窗口(Help Navigator / Browser)

单击工具栏的 ? 图标；或选择菜单“View”→“Help”；或选择菜单“Help”→“MATLAB Help”都能出现帮助导航 / 浏览器窗口。

9. 程序性能剖析窗口(Profiler)

▪ 选择菜单“View”→“Profiler”；或在命令窗口输入“profile viewer”命令都可以独立出现程序性能剖析窗口，如图 1.18 所示

▪ 使用菜单“View”→“Dock profiler”命令将该窗口放到 MATLAB 的操作界面中。

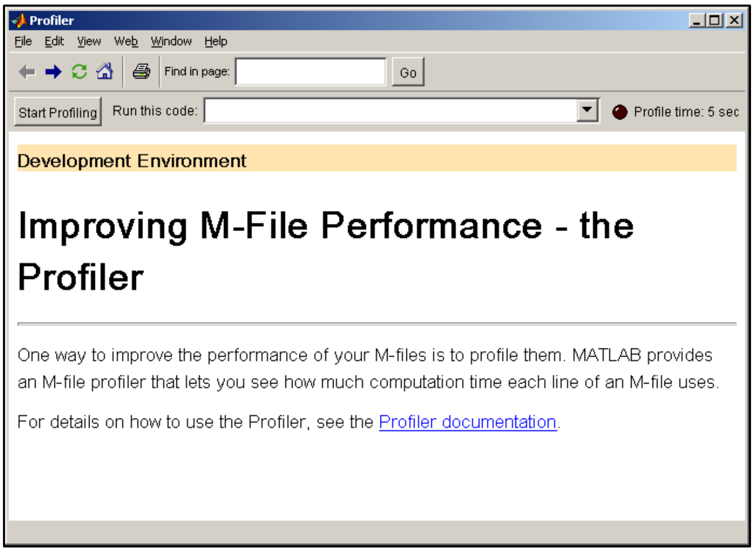


图 1.18 程序性能剖析窗口

1.3 MATLAB 6.5 帮助

MATLAB6.5 的帮助方式有很多种，用户可以通过快捷方便的帮助系统来迅速掌握 MATLAB 的强大功能。

1. 帮助导航 / 浏览器窗口

通过上节介绍的方法打开帮助导航 / 浏览器窗口，如图 1.19 所示。

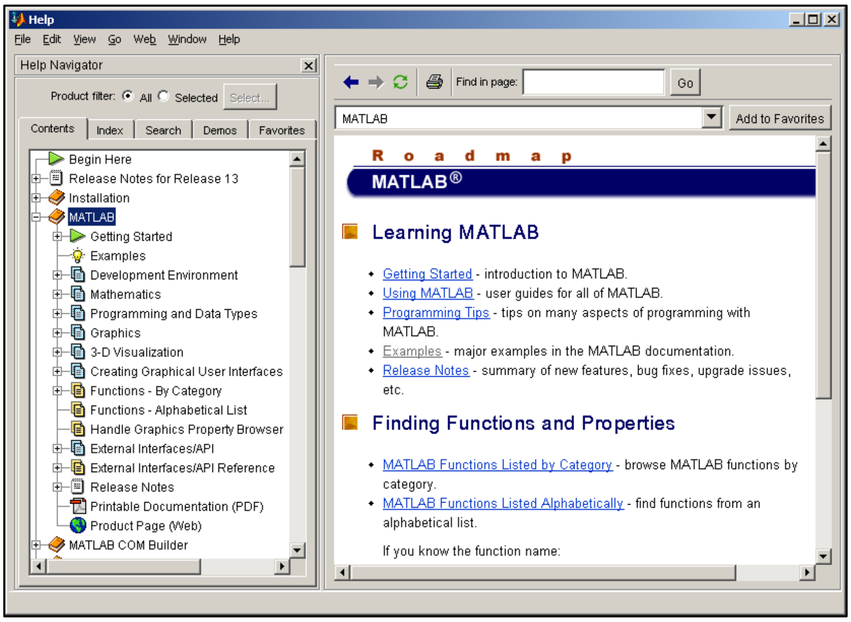


图 1.19 帮助导航 / 浏览器窗口

图 1.19 所示的帮助导航 / 浏览器窗口界面由左侧的 Help Navigator(帮助导航器)和右侧的 Help Browser(帮助浏览器)两部分组成。

帮助导航器的功能：

(1) Contents 选项窗口

- “Begin Here”是主要简介 MATLAB 的特点、内容和方法。
- “Release Notes For Release R13” 是专门介绍版本升级的变化。
- “Installation”是介绍各种环境下的安装方法。
- “MATLAB”下的各条目是最常使用的。

“Getting Started”是对 MATLAB 的环境、图形和编程进行简单介绍；

“Examples”则是较全面进行举例；

“Development Environment”介绍了 MATLAB 的工作环境，有较综合的计算实例；

“Mathematics ” 是详细介绍 MATLAB 的数学运算；

“Programming and Data Types”介绍 M 文件编程和数据类型；

“Graphics”介绍绘图功能和图形用户界面设计；

“Printable Documentation”则是给出可打印的 PDF 文件列表等等。

(2) Index 选项窗口

Index 选项窗口是 MATLAB 提供的术语索引表，可以查找命令、函数和专用术语等。

(3) Search 选项窗口

Search 选项窗口是通过关键词来查找全文中与之匹配的章节条目。

(4) Demos 选项窗口

Demos 选项窗口用来运行 MATLAB 提供了 Demo。

(5) Favorites 选项窗口

Favorites 选项窗口罗列用户自己以前所做的读书标记(或称书签)，以供今后查阅方便。

2. 通过命令实现帮助

- **help** ：列出所有主要的帮助主题，每个帮助主题与 MATLAB 搜索路径的一个目录名相对应

help topic　　%给出指定主题的帮助，主题可以是函数、目录或局部路径

- **lookfor**：在所有的帮助条目中搜索关键字，常用来查找具有某种功能而不知道准确名字的命令。

lookfor topic　　% 把在搜索中发现与关键字相匹配的所有 M 文件的 H1 行(第一行注释)都显示出来

lookfor topic –all　%在所有 M 文件中搜索关键字

- **helpwin**：打开并显示帮助导航 / 浏览器窗口(如图 1.19 所示)。

helpwin topic　　%打开帮助导航 / 浏览器窗口显示指定的主题信息

3. PDF 帮助

MATLAB 6.5 把帮助导航 / 浏览器中的部分内容制作成了 PDF 文件，PDF 文件被分类存放在“..\matlab\help\pdf-doc”文件夹中。阅读这种文件需要 Adobe Acrobat Reader 软件支持。

4. 其他帮助

(1) Demos 演示

Demos 演示界面操作非常方便，为用户提供了图文并茂的演示实例。演示程序是一个很好的学习过程，可以作为对 MATLAB 功能的浏览。

(2) 通过 Web 查找帮助信息

MathWorks 公司提供了技术支持网站，通过该网站用户可以找到相关的 MATLAB 书籍介绍、MATLAB 使用建议、常见问题解答和其他 MATLAB 用户提供的应用程序等。

1.4　MATLAB 6.5 其他管理

1.4.1 MATLAB 用户文件格式

1. 程序文件

程序文件即 M 文件，其文件的扩展名为.m，包括主程序和函数文件，M 文件通过 M 文件编辑 / 调试器生成。MATLAB 的各工具箱中的函数大部分是 M 文件。

2. 数据文件

数据文件即 MAT 文件，其文件的扩展名为.mat，用来保存工作空间的数据变量，数据文件可以通过在命令窗口中输入“save”命令生成。

3. 可执行文件

可执行文件即 MEX 文件，其文件的扩展名为.mex，由 MATLAB 的编译器对 M 文件进行编译后产生，其运行速度比直接执行 M 文件快得多(在 8.1 小节介绍)。

4. 图形文件

图形文件的扩展名为.fig，可以在“File”菜单中创建和打开，也可由 MATLAB 的绘图命令和图形用户界面窗口产生。

5. 模型文件

模型文件扩展名为.mdl，是由 Simulink 工具箱建模生成的。另外，还有仿真文件.s 文件。

1.4.2 设置搜索路径

1. MATLAB 的基本搜索过程

MATLAB 按照以下步骤进行搜索：

- 在 MATLAB 内存中进行检查，检查 X 是否为工作空间的变量或特殊变量；
- 检查 X 是否为 MATLAB 的内部函数(Built-in Function)；
- 在当前目录上，检查是否有名为“X.m”或“X.mex”的文件存在；
- 在 MATLAB 搜索路径的所有其他目录中，检查是否有名为“X.m”或 “X.mex”的文件存在；
- 如果都不是，则 MATLAB 发出错误信息。

注意：

命令“exist”、“which”和“load”执行时也都遵循 MATLAB 搜索步骤的先后次序。

2. MATLAB 搜索路径的扩展和修改

当用户的某些目录不在搜索路径上，必须修改搜索路径。

(1) 利用设置路径对话框修改搜索路径

- 在 MATLAB 界面选择菜单 “File” → “Set Path” 命令。
- 在命令窗口运行 “pathtool” 命令。

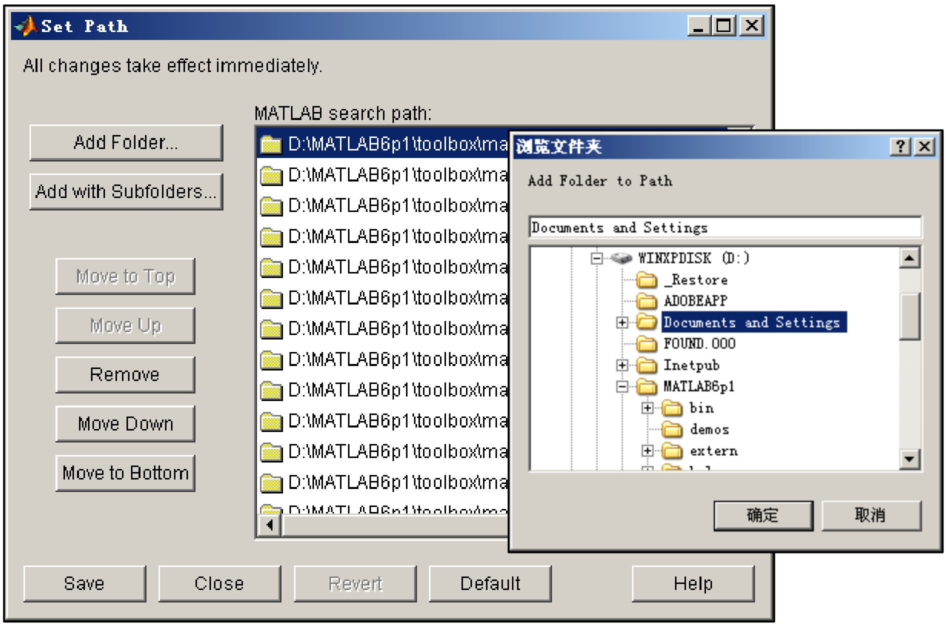


图 1.21 设置路径对话框

通过 “Add Folder…” 和 “Add with Subfolders…” 按钮打开浏览文件夹窗口来添加搜索目录。

(2) 利用 path 命令设置搜索路径

path (path, '新增目录')　　%在 MATLAB 的搜索路径的末尾添加新目录

```
>> path(path,'c: :\MyDir ')
```

1.4.3 文件管理命令

- **what** ：列出当前目录下的 M、MAT、MEX 文件清单

```
>> what
M-files in the current directory D:\MATLAB6p1\toolbox\matlab\elfun
Contents     asech      cosh       isreal     sec
abs          asin       cot        log        sech
acos         asinh      coth       log10      sign
acosh        atan       cplxpair   log2       sin
acot         atan2      csc        mod        sinh
acoth        atanh      csch       nextpow2   sqrt
acsc         ceil       exp        pow2       tan
acsch        complex    fix        real       tanh
angle        conj       floor      rem        unwrap
asec         cos        imag       round
MEX-files in the current directory D:\MATLAB6p1\toolbox\matlab\elfun
complex
```

- **dir**：列出指定目录下的文件和子目录清单

dir 目录名　%列出指定目录下的文件和子目录清单

```
>> dir
.              asinh.m        cplxpair.m   pow2.m
..             atan.m         csc.m        real.m
Contents.m     atan2.m        csch.m       rem.m
abs.m          atanh.m        exp.m        round.m
acos.m         ceil.m         fix.m        sec.m
acosh.m        complex.c      floor.m      sech.m
acot.m         complex.csf    imag.m       sign.m
acoth.m        complex.dll    isreal.m     sin.m
acsc.m         complex.m      ja           sinh.m
acsch.m        conj.m         log.m        sqrt.m
angle.m        cos.m          log10.m      tan.m
asec.m         cosh.m         log2.m       tanh.m
asech.m        cot.m          mod.m        unwrap.m
asin.m         coth.m         nextpow2.m
```

- **type 文件名**： 显示指定 M 文件的内容

```
>> type abs.m
%ABS     Absolute value.
%    ABS(X) is the absolute value of the elements of X. When
%    X is complex, ABS(X) is the complex modulus (magnitude) of
%    the elements of X.
%
%    See also SIGN, ANGLE, UNWRAP.
%    Copyright 1984-2001 The MathWorks, Inc.
%    $Revision: 5.8 $   $Date: 2001/04/15 12:02:51 $
%    Built-in function.
```

- **which 文件名** ：指出 M 文件、MEX 文件、工作空间变量、内置函数或 Simulink 模型所在的目录

```
>> which abs.m
D:\MATLAB6p1\toolbox\matlab\elfun\abs.m
```

- **matlabroot** ：返回安装 MATLAB 的根目录

```
>> matlabroot
ans =
D:\MATLAB6p1
```

- **diary**：把当前命令窗口中的所有内容(包括命令、计算结果等)保存到日志文件中

  diary ('file')　　　%使用指定文件名创建日志文件

  diary off　　　　　%暂停执行 diary 命令

  diary on　　　　　%恢复执行 diary 命令并使用当前的文件名

1.4.4 退出 MATLAB

要想退出 MATLAB 环境，可以使用以下任何一种方式：

- 在 MATLAB 的命令窗口输入“exit”命令。
- 在 MATLAB 的命令窗口输入“quit”命令。
- 直接单击 MATLAB 的命令窗口的按钮。

1.5 一个实例

【例 1.3】在 MATLAB 的通用操作界面综合地作一个练习。

- 启动 MATLAB。
- 在命令窗口(Command Window)中输入以下几行命令：

```
a=[1 2 3; 4 5 6;7 8 9];
b=[1 1 1;2 2 2;3 3 3 ];
c='计算';
d=a+b*i
```

- 打开工作空间浏览器窗口(Workspace Browser)查看变量。

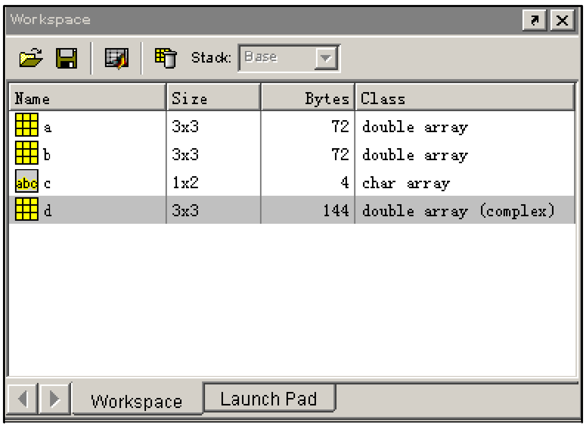


图 1.22 工作空间窗口

- 双击其中的变量“d”，出现数组编辑器窗口(Array Editor)，图 1.23 显示了该变量的

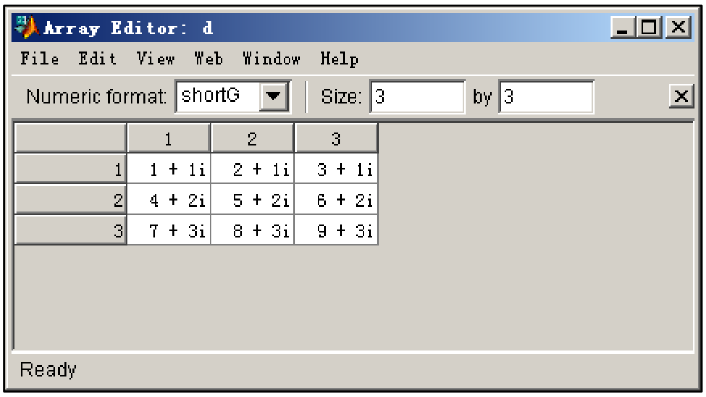


图 1.23 数组编辑器窗口

详细信息。

▪ 打开历史命令窗口(Command History)，如图 1.24 所示，选择上面的四行命令，单击鼠标右键在快捷菜单中选择“Create M-File”命令生成 M 文件。

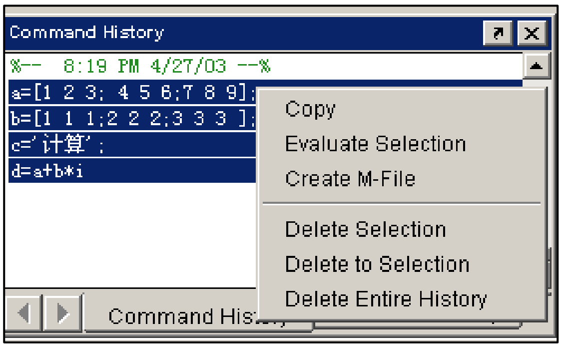


图 1.24 历史命令窗口

▪ 出现 M 文件编辑 / 调试器窗口(Editor / Debugger)，如图 1.25 所示。选择工具栏的“Save”按钮，将该文件保存为“c:\MyDir\Ex0103.m”。

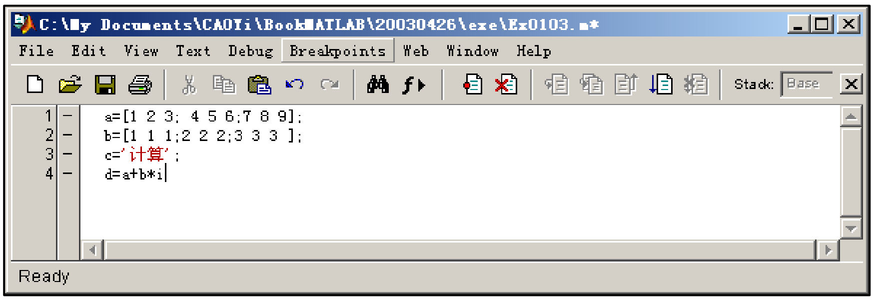


图 1.25 M 文件编辑 / 调试器窗口

▪ 打开当前目录浏览器窗口(Current Directory Browser)，将当前目录设置为“c:\MyDir”，可以看到刚保存的“Ex0103.m”文件，在命令窗口输入“Ex0103”运行该文件。

▪ 在命令窗口输入“save Ex0103”命令，在当前目录浏览器窗口可以看到在当前目录下生成了一个“Ex0103.mat”数据文件，如图 1.26 所示。

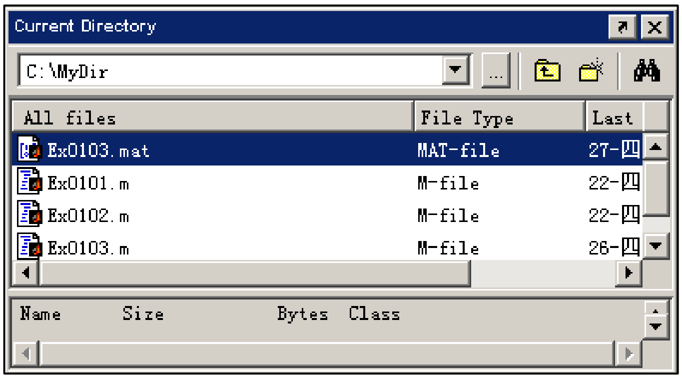


图 1.26 当前目录浏览器窗口

- 在命令窗口输入“exit”命令退出 MATLAB。
- 重新启动 MATLAB 后，在命令窗口输入“Ex0103”则不能运行该文件，因为该文件不在 MATLAB 的搜索路径中。

单击 MATLAB 界面的菜单“File”→“Set Path”，打开设置路径对话框，将“c:\MyDir”目录添加到搜索路径中，重新输入“Ex0103”则可以运行该文件。

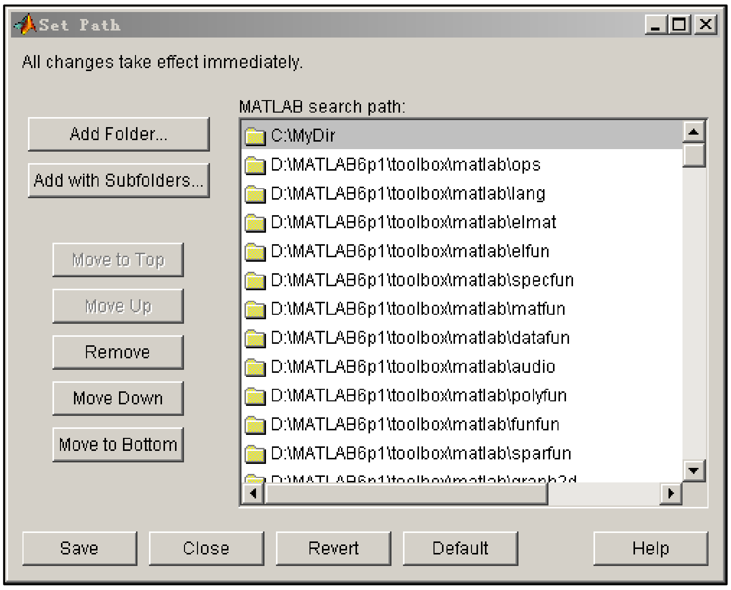


图 1.27 设置路径对话框

- 将“Ex0103.mat”数据文件的变量导入，可选择菜单“File”→“Import Data”命令，然后选择“c:\MyDir\ Ex0103.mat”文件。

```
>> type Ex0103
a=[1 2 3; 4 5 6;7 8 9];
b=[1 1 1;2 2 2;3 3 3 ];
c='计算';
d=a+b*i
```

第 2 章　MATLAB 数值计算

MATLAB 的数学计算＝数值计算＋符号计算

其中符号计算是指使用未定义的符号变量进行运算，而数值计算不允许使用未定义的变量。

2.1 变量和数据

2.1.1 数据类型

数据类型包括：数值型、字符串型、元胞型、结构型等

数值型＝双精度型、单精度型和整数类

整数类＝无符号类(uint8、uint16、uint32、uint64)和符号类整数(int8、int16、int32、int64)。

2.1.2 数据

1. 数据的表达方式

- 可以用带小数点的形式直接表示
- 用科学计数法
- 数值的表示范围是 10^{-309}～10^{309}。

以下都是合法的数据表示：

-2、5.67、2.56e-56(表示 2.56×10-56)、4.68e204(表示 4.68×10204)

2. 矩阵和数组的概念

在 MATLAB 的运算中，经常要使用标量、向量、矩阵和数组，这几个名称的定义如下：

- 标量：是指 1×1 的矩阵，即为只含一个数的矩阵。
- 向量：是指 1×n 或 n×1 的矩阵，即只有一行或者一列的矩阵。
- 矩阵：是一个矩形的数组，即二维数组，其中向量和标量都是矩阵的特例，0×0 矩阵为空矩阵([])。
- 数组：是指 n 维的数组，为矩阵的延伸，其中矩阵和向量都是数组的特例。

3. 复数

复数由实部和虚部组成，MATLAB 用特殊变量“i”和“j”表示虚数的单位。复数运算不需要特殊处理，可以直接进行。

复数可以有几种表示：

z=a+b*i 或 z=a+b*j

z=a+bi 或 z=a+bj(当 b 为标量时)

z=r*exp(i*theta)

- 得出一个复数的实部、虚部、幅值和相角。

```
a=real(z)          %计算实部
b=imag(z)      %计算虚部
```

r=abs(z)　　　　　　　%计算幅值

theta=angle(z)　　%计算相角

说明：

复数 z 的实部 a=r*cos(θ)；

复数 z 的虚部 b=r*sin(θ)；

复数 z 的幅值 $r=\sqrt{a^2+b^2}$ ；

复数 z 的相角 theta=arctg(b/a)，以弧度为单位。

```
a=1-2*i

a =
   1.0000 - 2.0000i

real(a)

ans =
     1

imag(a)

ans =
    -2

abs(a)

ans =
    2.2361

angle(a)*180/pi            %以角度为单位计算相角

ans =
  -63.4349
```

2.1.3 变量

1. 变量的命名规则

- 变量名区分字母的大小写。例如，“a”和“A”是不同的变量。
- 变量名不能超过 63 个字符，第 63 个字符后的字符被忽略，对于 MATLAB6.5 版以前的变量名不能超过 31 个字符。
- 变量名必须以字母开头，变量名的组成可以是任意字母、数字或者下划线，但不能含有空格和标点符号(如，。%等)。例如，“6ABC”、“AB%C”都是不合法的变量名。
- 关键字(如 if、while 等)不能作为变量名。

2. 特殊变量

MATLAB 有一些自己的特殊变量，当 MATLAB 启动时驻留在内存。

表 2.1 特殊变量表

特殊变量	取值
ans	运算结果的默认变量名
pi	圆周率 π
eps	计算机的最小数
flops	浮点运算数
inf	无穷大，如 1/0
NaN 或 nan	非数，如 0/0、∞/∞、$0\times\infty$
i 或 j	$i=j=\sqrt{-1}$
nargin	函数的输入变量数目
nargout	函数的输出变量数目
realmin	最小的可用正实数
realmax	最大的可用正实数

- 在 MATLAB 中系统将计算的结果自动赋给名为“ans”的变量。

```
2*pi

ans =
    6.2832
```

2.2 矩阵和数组

MATLAB 最基本也是最重要的功能就是进行实数或复数矩阵的运算。

2.2.1 矩阵输入

(1) 矩阵元素应用方括号([])括住；
(2) 每行内的元素间用逗号或空格隔开；
(3) 行与行之间用分号或回车键隔开；
(4) 元素可以是数值或表达式。

1. 通过显式元素列表输入矩阵

```
c=[1 2;3 4;5 3*2]          % [ ]表示构成矩阵,分号分隔行,空格分隔元素

c =
     1     2
     3     4
     5     6
```

用回车键代替分号分隔行:

```
 c=[1 2
3 4
5 6]
     1     2
     3     4
     5     6
```

2. 通过语句生成矩阵

(1) 使用 from:step:to 方式生成向量

from:to

from:step:to

说明：

from、step 和 to 分别表示开始值、步长和结束值。

当 step 省略时则默认为 step=1；

当 step 省略或 step>0 而 from>to 时为空矩阵，当 step<0 而 from<to 时也为空矩阵。

【例 2.1】使用“from:step:to”方式生成以下矩阵。

```
 x1=2:5

x1 =
     2     3     4     5
 x2=2:0.5:4

x2 =
    2.0000    2.5000    3.0000    3.5000    4.0000
 x3=5:-1:2

x3 =
     5     4     3     2
 x4=2:-1:3              %空矩阵

x4 =
   Empty matrix: 1-by-0
 x5=2:-1:0.5

x5 =
     2     1
 x6=[1:2:5;1:3:7]          %两行向量构成矩阵

x6 =
     1     3     5
     1     4     7
```

(2) 使用 linspace 和 logspace 函数生成向量

linspace(a,b,n)

说明：

a、b、n 三个参数分别表示开始值、结束值和元素个数。

生成从 a 到 b 之间线性分布的 n 个元素的行向量，n 如果省略则默认值为 100。

- logspace 用来生成对数等分向量，它和 linspace 一样直接给出元素的个数而得出各个元素的值。

logspace (a,b,n)

说明：

a、b、n 三个参数分别表示开始值、结束值和数据个数，n 如果省略则默认值为 50。生成从 10^a 到 10^b 之间按对数等分的 n 个元素的行向量。

【例 2.2】用 linspace 和 logspace 函数生成行向量。

```
 x1=linspace(0,2*pi,5)       %从 0 到 2*pi 等分成 5 个点

x1 =
        0    1.5708    3.1416    4.7124    6.2832

 x2=logspace(0,2,3)          %从 1 到 100 对数等分成 3 个点

x2 =
     1    10   100
```

3. 由矩阵生成函数产生特殊矩阵

MATLAB 提供了很多能够产生特殊矩阵的函数，各函数的功能如表 2.2 所示。

表 2.2 矩阵生成函数

函数名	功能	例子	
		输入	结果
zeros(m,n)	产生 m×n 的全 0 矩阵	zeros(2,3)	ans = 0 0 0 0 0 0
ones(m,n)	产生 m×n 的全 1 矩阵	ones(2,3)	ans = 1 1 1 1 1 1
rand(m,n)	产生均匀分布的随机矩阵，元素取值范围 0.0～1.0。	rand(2,3)	ans = 0.9501 0.6068 0.8913 0.2311 0.4860 0.7621
randn(m,n)	产生正态分布的随机矩阵	randn(2,3)	ans = -0.4326 0.1253 -1.1465 -1.6656 0.2877 1.1909

magic(N)	产生 N 阶魔方矩阵(矩阵的行、列和对角线上元素的和相等)	`magic(3)`	ans = 8 1 6 3 5 7 4 9 2
eye(m,n)	产生 m×n 的单位矩阵	`eye(3)`	ans = 1 0 0 0 1 0 0 0 1

注意：

zeros、ones、rand、randn 和 eye 函数当只有一个参数 n 时，则为 n×n 的方阵;

当 eye(m,n)函数的 m 和 n 参数不相等时则单位矩阵会出现全 0 行或列。

【例 2.3】查看 eye 函数的功能。

```
X1=eye(2,3)

X1 =
     1     0     0
     0     1     0

X2=eye(3,2)

X2 =
     1     0
     0     1
     0     0
```

4. 通过 MAT 数据文件加载矩阵

通过“load”命令或选择菜单“File”→“Import Data”命令加载 MAT 数据文件来创建矩阵。

5. 在 M 文件中创建矩阵

M 文件实际上是一种包含 MATLAB 代码的文本文件;

通过在 MATLAB 命令窗口中运行 M 文件创建矩阵。

2.2.2 矩阵元素和操作

矩阵和多维数组都是由多个元素组成的，每个元素通过下标来标识。

1. 矩阵的下标

(1) 全下标方式

矩阵中的元素可以用全下标方式标识，即由行下标和列下标表示，一个 m×n 的 a 矩阵的第 i 行第 j 列的元素表示为 a(i,j)。

注意：

- 如果在提取矩阵元素值时，矩阵元素的下标行或列(i,j)大于矩阵的大小(m,n)，则 MATLAB 会提示出错；
- 而在给矩阵元素赋值时，如果行或列(i,j)超出矩阵的大小(m,n)，则 MATLAB 自动扩充矩阵，扩充部分以 0 填充。

```
 a=[1 2;3 4;5 6]

a =
    1    2
    3    4
    5    6
 a(3,3)                  %提取 a(3,3)的值

??? Index exceeds matrix dimensions.
 a(3,3)=9                    %给 a(3,3) 赋值

a =
    1    2    0
    3    4    0
    5    6    9
```

(2) 单下标方式

先把矩阵的所有列按先左后右的次序连接成"一维长列"，然后对元素位置进行编号。以 m×n 的矩阵 a 为例，若元素 a(i,j)则对应的"单下标"为 s= (i-1)×m＋j。

2. 子矩阵块的产生

子矩阵是从对应矩阵中取出一部分元素构成，用全下标和单下标方式取子矩阵。

(1) 用全下标方式

矩阵 a 为图 2.2 所示，则：

- 取行数为 1、3，列数为 2、3 的元素构成子矩阵。

```
a([1 3],[2 3])

ans =
    2    0
    6    9
```

- 取行数为 1～3，列数为 2～3 的元素构成子矩阵，"1：3" 表示 1、2、3 行下标。

```
a(1:3,2:3)

ans =
```

```
     2     0
     4     0
     6     9
```

- 取所有行数即为 1～3，列数为 3 的元素构成子矩阵，“：”表示所有行或列。

```
a(:,3)

ans =
     0
     0
     9
```

- 取行数为 1～3，列数为 3 的元素构成子矩阵，用“end”表示某一维数中的最大值，即 3。

```
a(1:3,end)

ans =
     0
     0
     9
```

(2) 用单下标方式

取单下标为 1、3、2、6 的元素构成子矩阵。

```
a([1 3;2 6])

ans =
     1     5
     3     6
```

(3) 逻辑矩阵

子矩阵也可以利用逻辑矩阵来标识；

逻辑矩阵是大小和对应矩阵相同，而元素值为 0 或者 1 的矩阵。

可以用 a(L1,L2)来表示子矩阵，其中 L1、L2 为逻辑向量，当 L1、L2 的元素为 0 则不取该位置元素，反之则取该位置的元素。

【例 2.5】利用逻辑矩阵来提取矩阵，其中矩阵 a 如上图 2.2 所示。

```
l1=logical([1 0 1])        %给出逻辑向量 l1

l1 =
     1     0     1
l2=logical([1 1 0])        %给出逻辑向量 l2

l2 =
     1     1     0

a(l1,l2)                   %取出 1、3 行且 1、2 列的元素

ans =
```

```
     1     2
     5     6
```

【例 2.5 续】逻辑矩阵可以由矩阵进行逻辑运算得出。

```
b=a>1              %得出逻辑向量 b

b =
     0     1     0
     1     1     0
     1     1     1
a(b)                   %按单下标顺序排成长列

ans =
     3
     5
     2
     4
     6
     9
```

3. 矩阵的赋值

- 全下标方式：a(i,j)=b，给 a 矩阵的部分元素赋值则 b 矩阵的行列数必须等于 a 矩阵的行列数。

```
clear a
a(1:2,1:3)=[1 1 1;1 1 1]          %给第一、二行元素赋值为全 1

a =
     1     1     1
     1     1     1
```

- 单下标方式：a(s)=b，b 为向量，元素个数必须等于 a 矩阵的元素个数。

```
a(5:6)=[2 3]                %给第 5、6 元素赋值

a =
     1     1     2
     1     1     3
```

- 全元素方式：a(:)=b，给 a 矩阵的所有元素赋值则 b 矩阵的元素总数必须等于 a 矩阵的元素总数，但行列数不一定相等。

```
a=[1 2;3 4;5 6]

a =
     1     2
     3     4
     5     6
```

```
b=[1 2 3;4 5 6]

b =
     1     2     3
     4     5     6

a(:)=b                    %按单下标方式给 a 赋值

a =
     1     5
     4     3
     2     6
```

4. 矩阵元素的删除

删除操作就是简单地将其赋值为空矩阵(用[]表示)。

```
a=[1 2 0;3 4 0;5 6 9]

a =
     1     2     0
     3     4     0
     5     6     9

a(:,3)=[]                    %删除一列元素

a =
     1     2
     3     4
     5     6

a(1)=[]                    %删除一个元素，则矩阵变为行向量

a =
     3     5     2     4     6

a=[]                    %删除所有元素为空矩阵

a =
     []
```

5. 生成大矩阵

在 MATLAB 中，可以通过方括号“[]”实现将小矩阵联接起来生成一个较大的矩阵。

```
a=[1 2 0;3 4 0;5 6 9]

a =
```

```
     1     2     0
     3     4     0
     5     6     9
```

```
[a;a]                    %联接成 6×3 的矩阵
```

```
ans =
     1     2     0
     3     4     0
     5     6     9
     1     2     0
     3     4     0
     5     6     9
```

```
a=[1 2 0;3 4 0;5 6 9]
[a a]                    %联接成 3×6 的矩阵
```

```
ans =
     1     2     0     1     2     0
     3     4     0     3     4     0
     5     6     9     5     6     9
```

```
a=[1 2 0;3 4 0;5 6 9]
[a(1:2,1:2) 10*a(1:2,2:3)]     %计算并联接
```

```
ans =
     1     2    20     0
     3     4    40     0
```

6. 矩阵的翻转

a =

```
     1     2     0
     3     4     0
     5     6     9
```

表 2.3 常用矩阵翻转函数

函数名	功能	例子	
		输入	结果
triu(X)	产生 X 矩阵的上三角矩阵，其余元素补 0。	`triu(a)`	`ans =` `1 2 0` `0 4 0` `0 0 9`
tril(X)	产生 X 矩阵的下三角矩阵，其余元素补	`tril(a)`	`ans =` `1 0 0`

	0。		3 4 0 5 6 9
flipud(X)	使矩阵 X 沿水平轴上下翻转	`flipud(a)`	ans = 5 6 9 3 4 0 1 2 0
fliplr(X)	使矩阵 X 沿垂直轴左右翻转	`fliplr(a)`	ans = 0 2 1 0 4 3 9 6 5
flipdim(X,dim)	使矩阵 X 沿特定轴翻转。dim=1，按行维翻转； dim=2，按列维翻转。	`flipdim(a,1)`	ans = 5 6 9 3 4 0 1 2 0
rot90(X)	使矩阵 X 逆时针旋转 900	`rot90(a)`	ans = 0 0 9 2 4 6 1 3 5

2.2.3 字符串

在 MATLAB 中，字符串是作为字符数组来引入的；
一个字符串由多个字符组成，用单引号(’’)来界定；
字符串是按行向量进行存储的，每一字符(包括空格)是以其 ASCII 码的形式存放。

```
clear
 str1='Hello'
```

```
str1 =
Hello
```

```
 str2='I like ''MATLAB'''      %重复单引号来输入含有单引号的字符串
```

```
str2 =
I like 'MATLAB'
```

```
 str3='你好!'                  %支持中文
```

```
str3 =
你好!
```

1. 字符串占用的字节

```
whos
```

```
  Name      Size                    Bytes  Class

  str1      1x5                        10  char array
  str2      1x15                       30  char array
  str3      1x3                         6  char array

Grand total is 23 elements using 46 bytes
```

2. 字符串函数

- length：用来计算字符串的长度(即组成字符的个数)。
- double：用来查看字符串的 ASCII 码储存内容，包括空格(ASCII 码为 32)。
- char：用来将 ASCII 码转换成字符串形式。
- class 或 ischar：用来判断某一个变量是否为字符串。class 函数返回 char 则表示为字符串，而 ischar 函数返回 1 表示为字符串。
- strcmp(x,y)：比较字符串 x 和 y 的内容是否相同。返回值如果为 1 则相同，为 0 则不同。
- findstr(x,x1)：寻找在某个长字符串 x 中的子字符串 x1，返回其起始位置。
- deblank(x)：删除字符串尾部的空格。

由于 MATLAB 将字符串以其相对应的 ASCII 码储存成一个行向量，因此如果字符串直接进行数值运算，则其结果就变成一般数值向量的运算，而不再是字符串的运算。

```
length(str1)                %字符串长度

ans =
     5

x1=double(str1)                %查看字符串的 ASCII 码

x1 =
    72   101   108   108   111
x2=str1+1                   %字符串的数值运算

x2 =
    73   102   109   109   112

char(x1)                    %将 ASCII 码转换成字符串形式

ans =
Hello
char(x2)
```

```
ans =
Ifmmp
```

```
 class(str1)                          %判断变量类型
```

```
ans =
char
```

```
 class(x1)
```

```
ans =
double
```

```
 ischar(str1)
```

```
ans =
     1
```

3. 使用一个变量来储存多个字符串

(1) 多个字符串组成一个新的行向量

将多个字符串变量直接用“,”连接，构成一个行向量，就可以得到一个新字符串变量。

```
clear
 str1='Hello';
 str2='I like ''MATLAB''';
 str3='你好!'
 str4=[str1,'! ',str2]            %多个字符串并排成一个行向量
```

```
str4 =
Hello! I like 'MATLAB'
```

(2) 使用二维字符数组

将每个字符串放在一行，多个字符串可以构成一个二维字符数组，但必须先在短字符串结尾补上空格符，以确保每个字符串(即每一行)的长度一样。否则 MATLAB 会提示出错：

```
 str5=[str1;str3]
```

```
??? Error using ==> vertcat
All rows in the bracketed expression must have the same
number of columns.
```

```
 str5=[str1;str3,'  ']                %将 str3 添加两个空格
```

```
str5 =
Hello
你好!
```

(3) 使用 str2mat、strvcat 和 char 函数

使用专门的 str2mat、strvcat 和 char 函数可以构造出字符串矩阵，而不必考虑每行的字符数是否相等，总是按最长的设置，不足的末尾用空格补齐。

```
str6=str2mat(str1,str2,str3)

str6 =
Hello
I like 'MATLAB'
你好！
```

```
str7=char(str1,str2,str3)

str7 =
Hello
I like 'MATLAB'
你好！
```

```
str8=strvcat(str1,str2)

str8 =
Hello
I like 'MATLAB'
```

```
whos

  Name       Size                    Bytes  Class

  str1       1x5                        10  char array
  str2       1x15                       30  char array
  str3       1x3                         6  char array
  str4       1x22                       44  char array
  str5       2x5                        20  char array
  str6       3x15                       90  char array
  str7       3x15                       90  char array
  str8       2x15                       60  char array

Grand total is 186 elements using 350 bytes
```

5. 执行字符串

如果需要直接“执行”某一字符串，可以使用 eval 命令，效果就如同直接在 MATLAB 命令窗口内输入此命令。

```
str9='a=2*5'

str9 =
a=2*5
```

```
eval(str9)                        %执行字符串
```

```
a =
    10
```

6. 显示字符串

字符串可以直接使用 disp 命令显示出来，即使后面加分号(;)也显示。

```
disp('请输入 2*2 的矩阵 a')
```

```
请输入 2*2 的矩阵 a
```

```
disp(str1)
```

```
Hello
```

2.2.4 矩阵和数组运算

矩阵运算有明确而严格的数学规则，矩阵运算规则是按照线性代数运算法则定义的；数组运算是按数组的元素逐个进行的。

1. 矩阵运算的函数

```
a =
     1     2     3
     4     5     6
     7     8     9
```

表 2.4 常用矩阵运算函数

函数名	功能	例子	
		输入	结果
det(X)	计算方阵行列式	`det(a)`	ans = 0
rank(X)	求矩阵的秩，得出的行列式不为零的最大方阵边长。	`rank(a)`	ans = 2
inv(X)	求矩阵的逆阵，当方阵 X 的 det(X)不等于零，逆阵 X-1 才存在。X 与 X-1 相乘为单位矩阵。	`inv(a )`	Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 1.541976e-018. ans =

			1.0e+016 * -0.4504 0.9007 -0.4504 0.9007 -1.8014 0.9007 -0.4504 0.9007 -0.4504
[v,d]=eig(X)	计算矩阵特征值和特征向量。如果方程Xv=vd存在非零解，则v为特征向量，d为特征值。	**[v,d]=eig(a)**	v = -0.2320 -0.7858 0.4082 -0.5253 -0.0868 -0.8165 -0.8187 0.6123 0.4082 d = 16.1168 0 0 0 -1.1168 0 0 0 -0.0000
diag(X)	产生X矩阵的对角阵	**diag(a)**	ans = 1 5 9
[l,u]=lu(X)	方阵分解为一个准下三角方阵和一个上三角方阵的乘积。l为准下三角阵，必须交换两行才能成为真的下三角阵。	**[l,u]=lu(a)**	l = 0.1429 1.0000 0 0.5714 0.5000 1.0000 1.0000 0 0 u = 7.0000 8.0000 9.0000 0 0.8571 1.7143 0 0 0.0000
[q,r]=qr(X)	m×n阶矩阵X分解为一个正交方阵q和一个与X同阶的上三角矩阵r的乘积。方阵q的边长为矩阵X的n和m中较小者，且其行列式的值为1。	**[q,r]=qr(a)**	q = -0.1231 0.9045 0.4082 -0.4924 0.3015 -0.8165 -0.8616 -0.3015 0.4082 r = -8.1240 -9.6011 -11.0782 0 0.9045 1.8091 0 0 -0.0000
[u,s,v]=svd(X)	m×n阶矩阵X分解为三个矩阵的乘积，其中u,v为n×n阶和m×m阶正交方阵，s为m×n阶的对角阵，对角线上的元素就是矩阵X的奇异值，其长度为n和m中的较小者。	**[u,s,v]=svd(a)**	u = -0.2148 0.8872 0.4082 -0.5206 0.2496 -0.8165 -0.8263 -0.3879 0.4082 s = 16.8481 0 0 0 1.0684 0 0 0 0.0000 v = -0.4797 -0.7767 -0.4082 -0.5724 -0.0757 0.8165 -0.6651 0.6253 -0.4082

说明：

在上表中 det(a)=0 或 det(a)虽不等于零但数值很小接近于零，则计算 inv(a)时，其解的精度比较低，用条件数(求条件数的函数为 cond)来表示，条件数越大，解的精度越低，MATLAB 会提出警告："条件数太大，结果可能不准确"。

```
a=[1 2 3;4 5 6;7 8 9]
```

```
a =
     1     2     3
     4     5     6
     7     8     9
```

```
 inv(a)
```

```
Warning: Matrix is close to singular or badly scaled.
        Results may be inaccurate. RCOND = 1.541976e-018.
ans =
  1.0e+016 *
   -0.4504    0.9007   -0.4504
    0.9007   -1.8014    0.9007
   -0.4504    0.9007   -0.4504
```

2. 矩阵和数组的算术运算

(1) 矩阵和数组的加＋、减运算－

- A 和 B 矩阵必须大小相同才可以进行加减运算。
- 如果 A、B 中有一个是标量，则该标量与矩阵的每个元素进行运算。

(2) 矩阵和数组的乘法*运算

- 矩阵 A 的列数必须等于矩阵 B 的行数，除非其中有一个是标量。
- 数组的乘法运算符为".*"，表示数组 A 和 B 中的对应元素相乘。A 和 B 数组必须大小相同，除非其中有一个是标量。

```
x1=[1 2;3 4;5 6];
 x2=eye(3,2)
```

```
x2 =
     1     0
     0     1
     0     0
```

```
 x1+x2           %矩阵相加
```

```
ans =
     2     2
     3     5
     5     6
```

```
x1.*x2          %数组相乘
```

```
ans =
     1     0
     0     4
     0     0
```

```
x1*x2           %矩阵相乘x1列数不等于x2行数
```

```
??? Error using ==> *
Inner matrix dimensions must agree.
```

```
x3=eye(2,3)
```

```
x3 =
     1     0     0
     0     1     0
```

```
x1*x3           %矩阵相乘
```

```
ans =
     1     2     0
     3     4     0
     5     6     0
```

(3) 矩阵和数组的除法

- 矩阵运算符为“\”和“/”分别表示左除和右除。

A\B=A^{-1}*B

A/B=A*B^{-1}。

其中：A^{-1} 是矩阵的逆，也可用 inv(A)求逆矩阵。

- 数组的除法运算表达式

“A.\B”和“A./B”，分别为数组的左除和右除，表示数组相应元素相除。

A 和 B 数组必须大小相同，除非其中有一个是标量。

【例 2.12】已知方程组 $\begin{cases} 2x_1 - x_2 + 3x_3 = 5 \\ 3x_1 + x_2 - 5x_3 = 5 \\ 4x_1 - x_2 + x_3 = 9 \end{cases}$，用矩阵除法来解线性方程组。

解：将该方程变换成 AX=B 的形式。

其中：

$$A = \begin{bmatrix} 2 & -1 & 3 \\ 3 & 1 & -5 \\ 4 & -1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 5 \\ 5 \\ 9 \end{bmatrix}$$

```
A=[2 -1 3;3 1 -5;4 -1 1]
```

```
A =
     2    -1     3
     3     1    -5
     4    -1     1
```

```
 B=[5;5;9]
```

```
B =
     5
     5
     9
     X=A\B
```

```
X =
     2
    -1
     0
```

- 在线性方程组 A*X=B 中，m×n 阶矩阵 A 的行数 m 表示方程数，列数 n 表示未知数的个数。
- n=m，A 为方阵，A\B＝inv(A)*B。
- m＞n，是最小二乘解，X=inv(A'*A)*(A'*B)
- m＜，则是令 X 中的 n-m 个元素为零的一个特殊解。X=inv(A'*A)*(A'*B)

(4) 矩阵和数组的乘方

- 矩阵乘方的运算表达式为"A^B"，其中 A 可以是矩阵或标量。

当 A 为矩阵，必须为方阵：

B 为正整数时，表示 A 矩阵自乘 B 次；

B 为负整数时，表示先将矩阵 A 求逆，再自乘|B|次，仅对非奇异阵成立；

B 为矩阵时不能运算，会出错；

B 为非整数时，将 A 分解成 A=W*D/W，D 为对角阵，则有 A^B=W*D^B/W。

当 A 为标量：

B 为矩阵时，将 A 分解成 A=W*D/W，D 为对角阵，则有 A^B=W*diag(D.^B)/W。

- 数组乘方的运算表达式"A.^B"。

当 A 为矩阵，B 为标量时，则将 A(i,j)自乘 B 次；

当 A 为矩阵，B 为矩阵时，A 和 B 数组必须大小相同，则将 A(i,j)自乘 B(i,j)次；

当 A 为标量，B 为矩阵时，将 A^ B(i,j)构成新矩阵的第 i 行第 j 列元素。

【**例 2.13**】矩阵和数组的除法和乘方运算。

```
 x1=[1 2;3 4];
 x2=eye(2)
```

```
x2 =
     1     0
     0     1
```

```
x1/x2                  %矩阵右除

ans =
     1     2
     3     4

inv(x1)                %求逆矩阵

ans =
   -2.0000    1.0000
    1.5000   -0.5000

x1\x2                  %矩阵左除

ans =
   -2.0000    1.0000
    1.5000   -0.5000

x1./x2                 %数组右除

Warning: Divide by zero.
(Type "warning off MATLAB:divideByZero" to suppress this warning.)
ans =
     1   Inf
   Inf     4

x1.\x2                 %数组左除

ans =
    1.0000         0
         0    0.2500

x1^2                       %矩阵乘方

ans =
     7    10
    15    22

x1^-1                  %矩阵乘方，指数为-1 与 inv 相同

ans =
   -2.0000    1.0000
    1.5000   -0.5000

x1^0.2                 %矩阵乘方，指数为小数
```

```
ans =
   0.8397 + 0.3672i   0.2562 - 0.1679i
   0.3842 - 0.2519i   1.2239 + 0.1152i
```

```
2^x1                        %标量乘方
```

```
ans =
   10.4827   14.1519
   21.2278   31.7106
```

```
2.^x1                    %数组乘方
```

```
ans =
     2     4
     8    16
```

```
x1.^x2                   %数组乘方
```

```
ans =
     1     1
     1     4
```

3. 矩阵和数组的转置

- 矩阵的转置运算

"A'"表示矩阵 A 的转置，如果矩阵 A 为复数矩阵，则为共轭转置。

- 数组的转置运算

"A.'"表示数组 A 的转置，如果数组 A 为复数数组，则不是共轭转置。

【例 2.14】矩阵和数组转置运算。

```
x1=[1 2;3 4];
x2=eye(2);
x3=x1+x2*i
```

```
x3 =
   1.0000 + 1.0000i   2.0000
   3.0000             4.0000 + 1.0000i
```

```
x3'                  %矩阵转置
```

```
ans =
   1.0000 - 1.0000i   3.0000
   2.0000             4.0000 - 1.0000i
```

```
x3.'                     %数组转置为共轭转置
```

```
ans =
   1.0000 + 1.0000i   3.0000
   2.0000             4.0000 + 1.0000i
```

4. 矩阵和数组的数学函数

MATLAB 中数学函数对数组的每个元素进行运算。数组的基本函数如表 2.5 所示。

表 2.5 基本函数

函数名	含义	函数名	含义
abs	绝对值或者复数模	rat	有理数近似
sqrt	平方根	mod	模除求余
real	实部	round	4 舍 5 入到整数
imag	虚部	fix	向最接近 0 取整
conj	复数共轭	floor	向最接近-∞取整
sin	正弦	ceil	向最接近-∞取整
cos	余弦	sign	符号函数
tan	正切	rem	求余数留数
asin	反正弦	exp	自然指数
acos	反余弦	log	自然对数
atan	反正切	log10	以 10 为底的对数
atan2	第四象限反正切	pow2	2 的幂
sinh	双曲正弦	bessel	贝赛尔函数
cosh	双曲余弦	gamma	伽吗函数
tanh	双曲正切		

【例 2.15】使用数组的算术运算函数。

```
t=linspace(0,2*pi,6)

t =
        0    1.2566    2.5133    3.7699    5.0265    6.2832

y=sin(t)                    %计算正弦

y =
        0    0.9511    0.5878   -0.5878   -0.9511   -0.0000

y1=abs(y)                   %计算绝对值，将正弦曲线变成全波整流

y1 =
        0    0.9511    0.5878    0.5878    0.9511    0.0000

1-exp(-t).*y                %计算按指数衰减的正弦曲线

ans =
    1.0000    0.7293    0.9524    1.0136    1.0062    1.0000
```

S 为标量，A、B 为矩阵。

表 2.6 矩阵和数组运算对比表

数组运算		矩阵运算	
命令	含义	命令	含义
A+B	对应元素相加	A+B	与数组运算相同
A-B	对应元素相减	A-B	与数组运算相同
S.*B	标量 S 分别与 B 元素的积	S*B	与数组运算相同
A.*B	数组对应元素相乘	A*B	内维相同矩阵的乘积
S./B	S 分别被 B 的元素左除	S\B	B 矩阵分别左除 S
A./B	A 的元素被 B 的对应元素除	A/B	矩阵 A 右除 B 即 A 的逆阵与 B 相乘
B.\A	结果一定与上行相同	B\A	A 左除 B(一般与上行不同)
A.^S	A 的每个元素自乘 S 次	A^S	A 矩阵为方阵时，自乘 S 次
A.^S	S 为小数时，对 A 各元素分别求非整数幂，得出矩阵	A^S	S 为小数时，方阵 A 的非整数乘方
S.^B	分别以 B 的元素为指数求幂值	S^B	B 为方阵时，标量 S 的矩阵乘方
A.’	非共轭转置，相当于 conj(A’)	A’	共轭转置
exp(A)	以自然数 e 为底，分别以 A 的元素为指数求幂	expm(A)	A 的矩阵指数函数
log(A)	对 A 的各元素求对数	logm(A)	A 的矩阵对数函数
sqrt(A)	对 A 的各元素求平方根	sqrtm(A)	A 的矩阵平方根函数
f(A)	求 A 各个元素的函数值	funm(A,’FUN’)	矩阵的函数运算

注意：

funm(A,’FUN’)要求 A 必须是方阵，“FUN”为矩阵运算的函数名。

5. 关系操作和逻辑操作

(1) 关系运算

关系操作符：

<、<=、>、>=、 = =(等于)、 ~=(不等于)

关系运算规则：

- 两个变量都是标量，则结果为真(1)或假(0)。
- 两个变量都是数组，则必须大小相同，结果也是同样大小的数组，数组的元素为 0 或 1。
- 一个数组和一个标量，则把数组的每个元素分别与标量比较，结果为与数组大小相同的数组，数组的元素为 0 或 1。
- <、<= 和 >、>=，仅对参加比较变量的实部进行比较，

= = 和 ~= ，则同时对实部和虚部进行比较。

(2) 逻辑运算

逻辑操作符：

&(与)、|(或)、~(非)和 xor(异或)。

&&(先决与)逻辑运算符是当该运算符的左边为 1(真)时，才继续执行该符号右边的运算。

|| (先决或)逻辑运算符是该当运算符的左边为 1(真)时，就不需要继续执行该符号右边的运算，而立即得出该逻辑运算结果为 1(真)；否则，就要继续执行该符号右边的运算。

逻辑运算规则：

▪ 在逻辑运算中，非 0 元素表示真(1)，0 元素表示假(0)，逻辑运算的结果为 0 或 1，逻辑运算法则如表 2.7 所示。

表 2.7　逻辑运算

a	b	a & b	a|b	~a	xor(a,b)
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

▪ 两个变量都是标量，则结果为 0、1 的标量。

▪ 两个变量都是数组，则必须大小相同，结果也是同样大小的数组。

▪ 一个数组和一个标量，则把数组的每个元素分别与标量比较，结果为与数组大小相同的数组。

```
a=0;b=5; c=10;
(a~=0)&&(b<c)
```

```
ans =
     0
```

```
(a~=0)||(b<c)
```

```
ans =
     1
```

【例 2.16】数组的关系和逻辑运算，其中 y1 曲线如图 2.3 所示。

```
t=linspace(0,3*pi,10);
y=sin(t)                %计算正弦曲线
```

```
y =
  Columns 1 through 6
         0    0.8660    0.8660    0.0000    -0.8660    -0.8660
  Columns 7 through 10
   -0.0000    0.8660    0.8660    0.0000
```

```
t1=(t<pi)|(t>2*pi)
```

```
t1 =
     1     1     1     0     0     0     0     1     1     1
```

```
y1=t1.*y                %得出 0～π 和 2π～3π 的半波整流
```

```
y1 =
```

```
  Columns 1 through 6
         0    0.8660    0.8660         0         0         0
  Columns 7 through 10
         0    0.8660    0.8660    0.0000
```

(3) 函数运算

MATLAB 中能得出真(1)和假(0)结果的函数有：

关系逻辑函数、工作状态判断函数、特殊数据判断函数和数据类型函数。

a =　　　　　　　　　　　　b=

　　1　　Inf　　　　　　　　　　0　　1

　　0　　2　　　　　　　　　　　1　　0

表2.8 关系逻辑函数

函数名	功能	例子	
		输入	结果
all(A)	判断 A 的列向量元素是否全非 0，全非 0 则为 1	all(a)	ans = 0 1
any(A)	判断 A 的列向量元素中是否有非 0 元素，有则为 1	any(a)	ans = 1 1
isequal(A,B)	判断A、B 对应元素是否全相等，相等为 1	isequal(a,b)	ans = 0
isempty(A)	判断 A 是否为空矩阵，为空则为 1，否则为 0	isempty(a)	ans = 0
isfinite(A)	判断 A 的各元素值是否有限，是则为 1	isfinite(a)	ans = 1 0 1 1
isinf(A)	判断 A 的各元素值是否无穷大，是则为 1	isinf(a)	ans = 0 1 0 0
isnan(A)	判断 A 的各元素值是否为NAN，是则为 1	isnan(a)	ans = 0 0 0 0
isnumeric(A)	判断数组 A 的元素是否全为数值型数组	isnumeric(a)	ans = 1
isreal(A)	判断数组 A 的元素是否全为实数，是则为 1	isreal(a)	ans = 1

isprime(A)	判断A的各元素值是否为质数，是则为1	isprime(b)	ans = 0 0 0 0
isspace(A)	判断A的各元素值是否为空格，是则为1	isspace(a)	ans = 0 0 0 0
find(A)	寻找A数组非0元素的下标和值	find(b)	ans = 2 3

6. 运算符优先级

在 MATLAB 中各种运算符的优先级如下：

'(矩阵转置)、^(矩阵幂)和.'(数组转置)、.^(数组幂)

→ ~(逻辑非)

→ *(乘)、/(左除)、\(右除)和.*(点乘)、./(点左除)、.\(点右除)

→+、-(加减)

→: (冒号)

→<、<=、>、>=、~=

→&(逻辑与)

→ |(逻辑或)

→ &&(先决与)

→ ||(先决或)

2.2.5 多维数组

1. 多维数组的创建

(1) 通过“全下标”元素赋值方式创建

```
a(:,:,2)=[1 2;3 4]          %创建三维数组
b=[1 1;2 2]                 %先创建二维数组
```

```
b =
     1     1
     2     2
```

```
b(:,:,2)=5                  %扩展数组
```

```
b(:,:,1) =
     1     1
     2     2
b(:,:,2) =
     5     5
```

```
     5     5
```

(2) 由函数 ones、zeros、rand 和 randn 直接创建

例如，用函数 rand 直接创建三维随机数组：

```
rand(2,4,3)

ans(:,:,1) =
    0.9501    0.6068    0.8913    0.4565
    0.2311    0.4860    0.7621    0.0185
ans(:,:,2) =
    0.8214    0.6154    0.9218    0.1763
    0.4447    0.7919    0.7382    0.4057
ans(:,:,3) =
    0.9355    0.4103    0.0579    0.8132
    0.9169    0.8936    0.3529    0.0099
```

(3) 利用函数生成数组

- 将一系列数组沿着特定的维连接成一个多维数组

语法：

cat(维,p1,p2,……)

说明：第一个参数维是指沿着第几维连接数组 p1、p2 等。

- 按指定行列数放置模块数组生成多维数组

语法：

repmat(p)

repmat(p,行 列 页 ……)

说明：第一个输入变量 p 是用来放置的模块数组，后面的变量是要放在指定的各维。

- 在总元素的数目不变的前提下重新确定数组的行列数来重组数组

语法：

reshape(p)

reshape(p,行 列 页 ……)

说明：第一个变量是待重组的数组 p，后面的变量是重新生成数组的行数、列数、页数”。

【例 2.18】用函数来生成多维数组。

```
a=[1 2;3 4];
b=[1 1;2 2];
c= cat(2,a,b)             %沿着第二维连接生成数组 c

c =
     1     2     1     1
     3     4     2     2
cat(3,a,b)                   %沿着第三维连接

ans(:,:,1) =
     1     2
     3     4
ans(:,:,2) =
     1     1
```

```
     2     2
 repmat(a,[2 2 2])        %放置模块数组 a

ans(:,:,1) =
     1     2     1     2
     3     4     3     4
     1     2     1     2
     3     4     3     4
ans(:,:,2) =
     1     2     1     2
     3     4     3     4
     1     2     1     2
     3     4     3     4
 reshape(c,[2 2 2])       %重组二维数组为 2 行 2 列 2 页的三维数组

ans(:,:,1) =
     1     2
     3     4
ans(:,:,2) =
     1     1
     2     2
```

2. 多维数组的标识

- 直接给出数组的维数

语法：

ndims (p)

- 给出数组各维的大小

语法：

[m,n,…]=size(p)　　　　%得出各维的大小

m=size(p,x)　　　　　　%得出某一维的大小

说明：p 为需要得出大小的多维数组；m 为行数，n 为列数…；当只有一个输出变量时，x=1 返回第一维(行数)，x=2 返回第二维(列数)，以此类推。

- 返回行数或列数的最大值

语法：

length(p)

说明：length(p)等价于 max(size(p))。

【例 2.19】得出矩阵的大小。

```
 a=[1 2;3 4;5 6]

a =
     1     2
     3     4
     5     6
 ndims(a)                 %得出维数
```

```
ans =
     2
 size(a)           %得出各维的大小

ans =
     3     2
 size(a,2)             %得出列的大小

ans =
     2
 length(a)             %得出最大维的大小

ans =
     3
```

2.3 稀疏矩阵

一个矩阵中如果包含很多元素值为 0，则此矩阵可以只存储少量的非 0 元素，这个矩阵称为稀疏矩阵(Sparse Matrix)。

2.3.1 稀疏矩阵的建立

稀疏矩阵大部分的元素都是 0，因此只需储存非零元素的下标和元素值，这种特殊的存储方式可以节省大量的存储空间和不必要的运算。

1. 使用 sparse 函数产生稀疏矩阵

sparse 函数用于创建稀疏矩阵，或将一个全元素矩阵直接转换成稀疏矩阵。

语法：

sparse(i,j,s,m,n)　　　%直接创建稀疏矩阵

sparse(p)　　　%由全元素矩阵 p 转换为稀疏矩阵

说明：i、j 是非 0 元素的行、列下标；s 是非 0 元素所形成的向量；m、n 是 s 的行、列维数，可省略；i、j、s 都是长度相同的向量，生成矩阵的元素 s(k)下标分别是 i(k)和 j(k)；p 为全元素矩阵。

【例 2.20】产生稀疏矩阵。

```
 a=eye(3);
 a(4,:)=[-5 -2 -3]

a =
     1     0     0
     0     1     0
     0     0     1
    -5    -2    -3
```

```
b=sparse(a)                                    %创建稀疏矩阵

b =
   (1,1)        1
   (4,1)       -5
   (2,2)        1
   (4,2)       -2
   (3,3)        1
   (4,3)       -3
```

```
c=sparse([1 4 2 4 3 4],[1 1 2 2 3 3],[1 -5 1 -2 1 -3])  %创建与 b 相同的稀疏矩阵

c =
   (1,1)        1
   (4,1)       -5
   (2,2)        1
   (4,2)       -2
   (3,3)        1
   (4,3)       -3
```

与 sparse 函数相反，full 函数可将稀疏矩阵转变为全元素矩阵。

语法：

full(p)　　　　　　　　　　　　%将稀疏矩阵 p 转变为全元素矩阵

【例 2.20 续】将稀疏矩阵转变为全元素矩阵。

```
full(b)

ans =
     1     0     0
     0     1     0
     0     0     1
    -5    -2    -3
```

2. 用 spdiags 函数创建稀疏矩阵

spdiags 函数是用对角线元素来构建一个稀疏矩阵。

语法：

spdiags(D,k,m,n)

说明：矩阵 D 的每一列代表矩阵的对角线向量；k 代表对角线的位置(0 代表主对角线，−1 代表向下位移一单位的次对角线，1 代表向上位移一单位的次对角线，依此类推)；m、n 分别代表矩阵的行、列维数。

【例 2.20 续】用 spdiags 函数创建稀疏矩阵。

```
D=[3 2 9;2 4 9;1 1 4]

D =
     3     2     9
     2     4     9
```

```
     1     1     4
d=[0 1 2];
s=spdiags(D,d,4,3)        %构成 4 行 3 列的稀疏矩阵

s =
   (1,1)        3
   (1,2)        4
   (2,2)        2
   (1,3)        4
   (2,3)        1
   (3,3)        1
full(s)

ans =
     3     4     4
     0     2     1
     0     0     1
     0     0     0
```

程序分析：可以看出矩阵 s 的三个非零对角线向量分别是 D 的三个列向量。主对角线为“3 2 1”；向上位移一单位的次对角线为“2 4 1”，但其中“2” 被移掉了；向上位移两单位的次对角线为“9 9 4”，但其中“9 9” 都被移掉了。

3. 用 spconvert 函数从外部文件输入稀疏矩阵

可以将 load 命令和 spconvert 函数结合，将文本文件(如*.dat)调用到内存，然后将文本文件内容转换成稀疏矩阵。也可以直接用 save 或 load 命令将稀疏矩阵保存到 MAT 文件中或从 MAT 文件调用到内存中。

语法：

spconvert (Filename)

【例 2.20 续】用 load 命令和 spconvert 函数结合转换稀疏矩阵。

用文本编辑器生成 ASCⅡ文本文件，spr.dat 的内容可显示如下：

1 1 1
4 1 -5
2 2 1
4 2 -2
3 3 1
4 3 -3

其中：第 1 列、第 2 列分别代表稀疏矩阵的行、列下标，第 3 列则是元素值。

```
load spr.dat              %装载文本文件

b= spconvert(spr)         %转换成稀疏矩阵
b =
   (1,1)        1
   (4,1)       -5
   (2,2)        1
   (4,2)       -2
   (3,3)        1
```

```
        (4,3)          -3
    save spr b                      %保存 spr.mat 文件
    clear
    load spr b                      %装载 spr.mat 文件
    b
   b =
        (1,1)           1
        (4,1)          -5
        (2,2)           1
        (4,2)          -2
        (3,3)           1
        (4,3)          -3
```

2.3.2 稀疏矩阵的存储空间

MATLAB 提供了得出稀疏矩阵元素个数的函数：

- nnz：可返回稀疏矩阵的非零元素个数。
- nonzeros：返回一个包含所有非零元素的列向量。
- nzmax：返回最大的非零元素个数，当 nnz>nzmax 时，MATLAB 会动态调整增加内存给 nzmax，以储存新增的非零元素。
- spy：用图形观看稀疏矩阵的非零元素分布情况。

【例 2.20 续】查看稀疏矩阵的非零元素，图 2.4 所示为用 spy 函数得出的稀疏矩阵分布图。

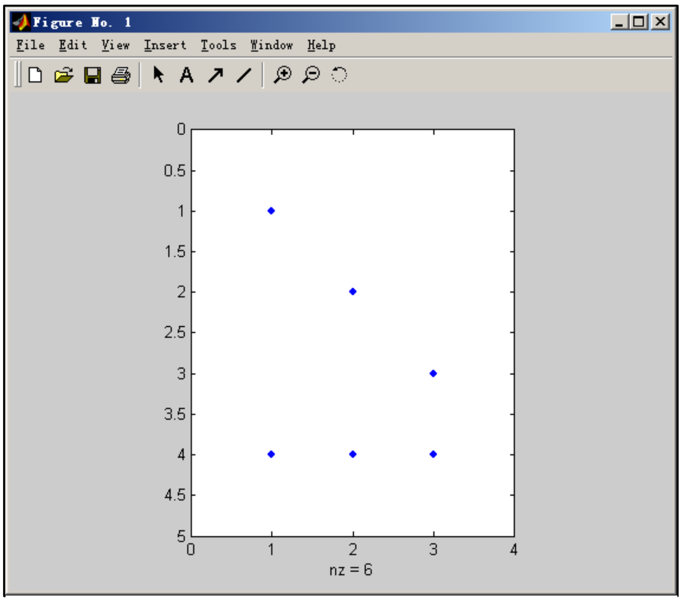


图 2.4 稀疏矩阵分布图

```
 nnz(b)                    %得出非零元素个数

ans =
     6
 nonzeros(b)               %得出非零元素
```

```
ans =
     1
    -5
     1
    -2
     1
    -3
 nzmax(b)

ans =
     6
 spy(b)
```

2.3.3 稀疏矩阵的运算

稀疏矩阵的标准数学运算按照以下原则：

- 如果函数的输入参数是向量或标量，输出的参数为矩阵，则输出参数为全元素矩阵；
- 如果函数的输入参数是矩阵，输出的参数为矩阵，则输出参数以输入矩阵的方式来表示，即当输入参数为稀疏矩阵时，输出参数也是稀疏矩阵；
- 如果二元运算的两个操作数中有一个是全元素矩阵一个是稀疏矩阵，则对于“+”、“-”、“*”、“\” 运算结果为全元素矩阵，而“&”、“.* ”运算结果为稀疏矩阵。
- 用“cat”函数或“[]”连接混合矩阵将产生稀疏矩阵。

2.4 多项式

例如，多项式 $p1(x)= x^3+21x^2+20x$ 可以表示为：

```
p1=[1 21 20 0]                    %常数项为 0
```

2.4.1 多项式的求值、求根和部分分式展开

1. 多项式求值

函数 polyval 可以用来计算多项式在给定变量时的值，是按数组运算规则进行计算的。

语法：

polyval(p,s)

说明：p 为多项式, s 为给定矩阵。

【例 2.21】计算 $p(x)= x^3+21x^2+20x$ 多项式的值。

```
p1=[1 21 20 0];
polyval(p1,2)          %计算 x=2 时多项式的值

ans =
   132
x=0:0.5:3;
```

```
polyval(p1,x)          %计算 x 为向量时多项式的值

ans =
         0   15.3750   42.0000   80.6250  132.0000  196.8750
276.0000
```

2. 多项式求根

- roots 用来计算多项式的根。

语法：

r=roots(p)

说明：p 为多项式；r 为计算的多项式的根，以列向量的形式保存。

- 与函数 roots 相反，根据多项式的根来计算多项式的系数可以用 poly 函数来实现。

语法：

p=poly (r)

【例 2.21 续】计算多项式 $p1(x)= x^3+21x^2+20x$ 的根以及由多项式的根得出系数。

```
 roots(p1)                 %计算多项式的根

ans =
     0
   -20
    -1
 poly([0;-20;-1])             %计算多项式的系数

ans =
     1    21    20     0
```

3. 特征多项式

对于一个方阵 s，可以用函数 poly 来计算矩阵的特征多项式的系数。特征多项式的根即为特征值，用 roots 函数来计算。

语法：

p=poly (s)

说明：s 必须为方阵；p 为特征多项式。

【例 2.21 续】根据矩阵来计算的特征多项式系数。

```
 s=[1 2;3 4]

s =
     1     2
     3     4
 p2=poly(s)              %计算特征多项式

p2 =
    1.0000   -5.0000   -2.0000
 roots(p2)                 %计算特征根
```

```
ans =
    5.3723
   -0.3723
```

程序分析：$p2=x^2-5x-2$ 为矩阵 s 的特征多项式，5.3723 和-0.3723 为矩阵 s 的特征根。也可以用 eig 函数来计算方阵 s 的特征值和特征向量的方法得出。

4. 部分分式展开

用 residue 函数来实现将分式表达式进行多项式的部分分式展开。

$$\frac{B(s)}{A(s)}=\frac{r_1}{s-p_1}+\frac{r_2}{s-p_2}+\cdots+\frac{r_n}{s-p_n}+k(s)$$

语法：

[r,p,k]=residue(b,a)

说明：b 和 a 分别是分子和分母多项式系数行向量；r 是[r_1 r_2 …r_n]留数行向量；p 为[p_1 p_2 …p_n]极点行向量；k 为直项行向量。

【例 2.21 续】将表达式 $\frac{100(s+2)}{s(s+1)(s+20)}$ 进行部分分式展开。

```
p1=[1 21 20 0];
p3=[100 200];
[r,p,k]=residue(p3,p1)
```

```
r =
   -4.7368
   -5.2632
   10.0000
p =
   -20
    -1
     0
k =
     []
```

程序分析：表达式 $\frac{100(s+2)}{s(s+1)(s+20)}$ 展开结果为 $\frac{-4.7368}{s+20}+\frac{-5.2632}{s+1}+\frac{10}{s}$。

2.4.2 多项式的乘除法和微积分

1. 多项式的乘法和除法

- 多项式的乘法

语法：

p=conv(pl,p2)

说明：p 是多项式 p1 和 p2 的乘积多项式。

- 多项式的除法

语法：

[q,r]=deconv(pl,p2)

说明：除法不一定会除尽，会有余子式。多项式 p1 被 p2 除的商为多项式 q，而余子式是 r。

【例 2.22】计算表达式 $s(s+1)(s+20)$。

```
a1=[1 0];                %对应多项式 s
a2=[1 1];                %对应多项式 s+1
a3=[1 20];               %对应多项式 s+20
p1=conv(a1,a2)

p1 =
     1     1     0
p1=conv(p1,a3)           %计算 s(s+1)(s+20)

p1 =
     1    21    20     0
[p2,r]=deconv(p1,a3)     %计算多项式除法的商和余子式

p2 =
     1     1     0
r =
     0     0     0     0
conv(p2,a3)+r            %用商*除式+余子式验算

ans =
     1    21    20     0
```

2. 多项式的微分和积分

- 多项式的微分由 polyder 函数实现。
- MATLAB 没有专门的多项式积分函数，但可以用[p./length(p):-1:1,k]的方法来完成积分， k 为常数。

【例 2.22 续】求多项式的微分和积分。

```
p4=polyder(p1)           %多项式微分

p4 =
     3    42    20
s=length(p4):-1:1

s =
     3     2     1
p1=[p4./s,0]             %多项式积分,常数 k=0

p1 =
     1    21    20     0
```

程序分析：可以看出多项式 $p4(x)= 3x^2+42x+20$ 的积分是 $p1(x)= x^3+21x^2+20x$。

2.4.3 多项式拟合和插值

1. 多项式拟合

多项式曲线拟合是用一个多项式来逼近一组给定的数据，使用 polyfit 函数来实现。拟合的准则是最小二乘法，即找出使 $\sum_{i=1}^{n}\left\|f(x_i)-y_i\right\|^2$ 最小的 f(x)。

语法：

p=polyfit(x,y,n)

说明：x、y 向量分别为 N 个数据点的横、纵坐标；n 是用来拟合的多项式阶次；p 为拟合的多项式，p 为 n+1 个系数构成的行向量。

【例 2.23】对多项式 $y_1=2x_1^3-x_1^2+5x_1+10$ 曲线拟合。经过一阶、二阶和三阶拟合的曲线如图 2.5 所示。

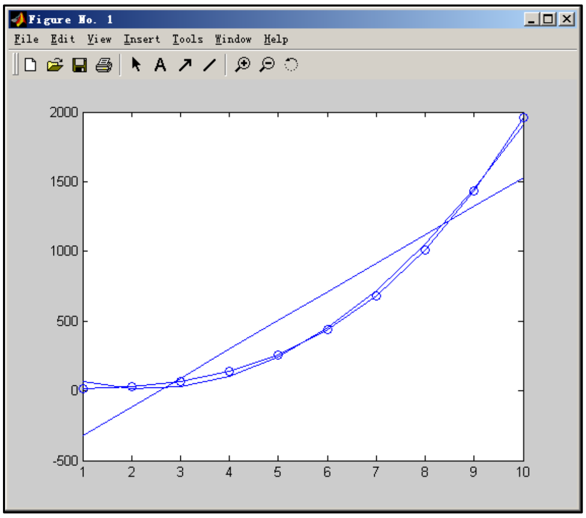


图 2.5 一阶、阶和三阶拟合曲线

```
x1=1:10;
p=[2 -1 5 10];
y0=polyval(p,x1)

y0 =
  Columns 1 through 6
        16          32          70         142         260         436
  Columns 7 through 10
       682        1010        1432        1960
p1=polyfit(x1,y0,1)              %一阶拟合

p1 =
  204.8000 -522.4000
p2=polyfit(x1,y0,2)              %二阶拟合

p2 =
```

```
  32.0000 -147.2000  181.6000
 p3=polyfit(x1,y0,3)              %三阶拟合

p3 =
    2.0000   -1.0000    5.0000   10.0000
```

2. 插值运算

插值运算是根据数据点的规律，找到一个多项式表达式可以连接两个点，插值得出相邻数据点之间的数值。

1. 一维插值

一维插值是指对一个自变量的插值，interp1 函数是用来进行一维插值的。

语法：

yi=interp1(x,y,xi,'method')

说明：x、y 为行向量；xi 是插值范围内任意点的 x 坐标，yi 则是插值运算后的对应 y 坐标；method 是插值函数的类型，"linear"为线性插值(默认)，"nearest"为用最接近的相邻点插值，"spline"为三次样条插值，"cubic"为三次插值。

【例 2.23 续】经过线性插值、三次样条插值计算出横坐标为 9.5 的对应纵坐标，如图 2.6 所示。

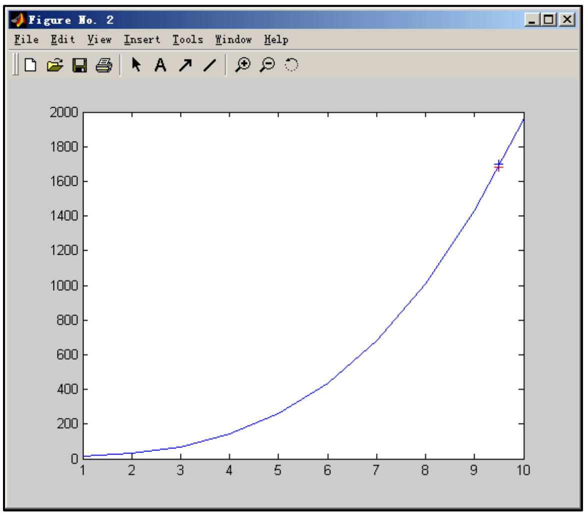


图 2.6 线性插值和三次样条插值

```
 x1
 y0
y01=interp1(x1,y0,9.5)           %线性插值

x1 =
    1     2     3     4     5     6     7     8     9    10
y0 =
  Columns 1 through 6
          16          32          70         142         260         436
  Columns 7 through 10
         682        1010        1432        1960
y01 =
```

```
      1696
 y02=interp1(x1,y0,9.5,'spline')       %三次样条插值

y02 =
      1682
```

2. 二维插值

二维插值是指对两个自变量的插值，interp2 函数是用来进行二维插值的。

语法：

zi=interp2(x,y,z,xi,yi,’method’)

说明：method 是插值函数的类型有，“linear”为双线性插值(默认)，“nearest”为用最接近点插值，“cubic”为三次插值。

2.5 元胞数组和结构数组

MATLAB 的元胞数组(Cell Array)和结构数组(Structure Array)都能在一个数组里存放各种不同类型的数据。

2.5.1 元胞数组

1. 元胞数组的创建

元胞数组中的基本组成是元胞，每一个元胞可以看成是一个单元(Cell)，用来存放各种不同类型的数据。

(1) 直接使用{}创建

【例 2.24】直接使用{}创建元胞数组。

```
 clear
A={'This is the first Cell.',[1 2;3 4];eye(3),{'Tom','Jane'}}

A =
   [1x23 char  ]    [2x2 double]
   [3x3  double]    {1x2 cell  }
 whos

  Name      Size                    Bytes  Class

  A        2x2                       524  cell array

Grand total is 49 elements using 524 bytes
```

程序分析：创建的元胞数组中的元胞 A(1,1)是字符串，A(1,2)是矩阵，A(2,1)是矩阵，而 A(2,2)为一个元胞数组。

(2) 由各元胞创建

【例 2.24 续】用创建各元胞的方法创建元胞数组。

```
 B(1,1)={'This is the second Cell.'}
```

```
B =
    'This is the second Cell.'
 B(1,2)={5+3*i}

B =
    [1x24 char]    [5.0000+ 3.0000i]
 B(1,3)={[1 2;3 4; 5 6]}

B =
    [1x24 char]    [5.0000+ 3.0000i]    [3x2 double]
```

(3) 由各元胞内容创建

【例 2.24 续】创建各元胞内容的方法创建元胞数组。

```
 C{1,1}='This is the third Cell.';
 C{2,1}=magic(4)

C =
    'This is the third Cell.'
    [4x4  double]
```

2. 元胞数组的内容显示

在 MATALB 命令窗口中输入元胞数组的名称，并不直接显示出元胞数组的各元素内容值，而是显示各元素的数据类型和维数。如【例 2.24】中显示元胞数组 A：

```
A

A =
    [1x23 char  ]    [2x2 double]
    [3x3  double]    {1x2 cell  }
```

(1) 使用 celldisp 命令显示元胞数组的内容

```
 celldisp(A)

A{1,1} =
This is the first Cell.
A{2,1} =
     1     0     0
     0     1     0
     0     0     1
A{1,2} =
     1     2
     3     4
A{2,2}{1} =
Tom
A{2,2}{2} =
Jane
 celldisp(B)
```

```
B{1} =
This is the second Cell.
B{2} =
   5.0000 + 3.0000i
B{3} =
     1     2
     3     4
     5     6
 celldisp(C)

C{1} =
This is the third Cell.
C{2} =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

程序分析：{}表示元胞数组的元胞元素内容，A{2,2}{1}表示第 2 行第 2 列的元胞元素中存放的元胞数组的第 1 个元胞元素的内容。

(2) 使用 cellplot 命令以图形显示元胞数组的内容

【例 2.24 续】用 cellplot 命令用图形显示元胞数组的内容，如图 2.7 所示。

```
 cellplot(A)
```

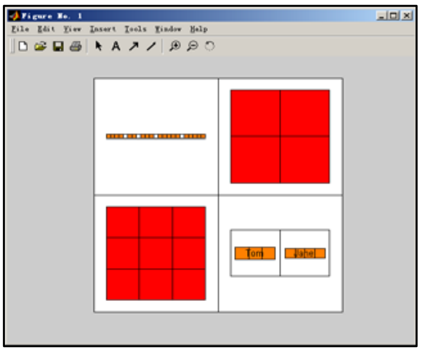


图 2.7 用图形显示元胞数组的内

3. 元胞数组的内容获取

(1) 取元胞数组的元素内容

【例 2.24 续】取出 A(1,2) 元胞元素的内容以及矩阵中的元素内容。

```
 x1=A{1,2}                %取 A(1,2) 元胞元素的内容

x1 =
     1     2
     3     4
 x2=A{1,2}(2,2)           %取 A(1,2) 元胞元素的矩阵第二行第二列内容
```

```
x2 =
     4
```

程序分析：x1 是矩阵，x2 是标量。

(2) 取元胞数组的元素

```
 x3=A(1,2)

x3 =
    [2x2 double]
```

程序分析：x3 是元胞数组。

(3) 使用 deal 函数取多个元胞元素的内容

```
 [x4,x5,x6]=deal(A{[2,3,4]})

x4 =
     1     0     0
     0     1     0
     0     0     1
x5 =
     1     2
     3     4
x6 =
    'Tom'    'Jane'
```

2.5.2 结构数组

结构数组的基本组成是结构(Structure)，每一个结构都包含多个域(Fields)。

1. 结构数组的创建

(1) 直接创建

【例 2.25】直接创建结构数组存放图形对象。

```
 ps(1).name='曲线 1'

ps =
    name: '曲线 1'
 ps(1).color='red'

ps =
     name: '曲线 1'
    color: 'red'
 ps(1).position=[0,0,300,300]

ps =
        name: '曲线 1'
       color: 'red'
    position: [0 0 300 300]
```

```
ps(2).name='曲线 2';
ps(2).color='blue';
ps(2).position=[100,100,300,300]

ps =
1x2 struct array with fields:
    name
    color
    position
```

程序分析：ps 是结构数组，ps(1)和 ps(2)是结构，name、color 和 position 是域。

(2) 利用 struct 函数创建

【例 2.25 续】利用 struct 函数创建结构数组。

```
ps(1)=struct('name','曲线1','color','red','position',[0,0,300,300]);
ps(2)=struct('name','曲线2','color','blue','position',[100,100,300,300])

ps =
1x2 struct array with fields:
    name
    color
    position
```

2. 结构数组数据的获取和设置

(1) 使用“.”符号获取

【例 2.25 续】结构数组数据的获取。

```
x1=ps(1)

x1 =
       name: '曲线 1'
      color: 'red'
   position: [0 0 300 300]
x2=ps(1).position

x2 =
     0     0   300   300
x3=ps(1).position(1,3)

x3 =
   300
```

程序分析：x1 是一个结构；x2 是矩阵；x3 是标量。

(2) 用 getfield 获取结构数组的数据

```
x4=getfield(ps,{1},'color')

x4 =
```

```
red
 x5=getfield(ps,{1},'color',{1})

x5 =
r
```

(3) 用 setfield 设置结构数组的数据

```
 ps=setfield(ps,{1},'color','green');
 ps(1)

ans =
       name: '曲线 1'
      color: 'green'
   position: [0 0 300 300]
```

3. 结构数组域的获取

(1) 使用 fieldnames 获取结构数组的所有域：

```
 x6=fieldnames(ps)

x6 =
    'name'
    'color'
    'position'
```

程序分析：x6 是元胞数组，各变量在工作空间的数据类型如图 2.8 所示。

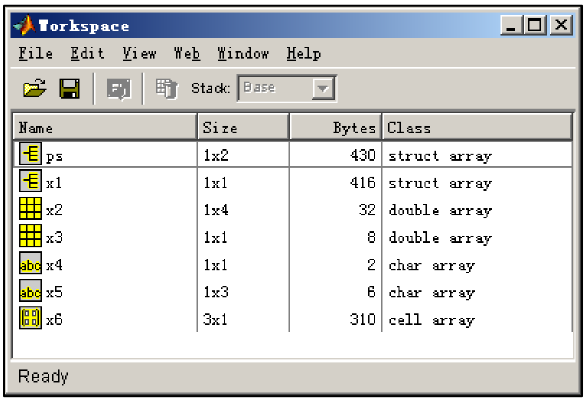


图 2.8 工作空间

(2) 获取结构数组域的数据

- 使用“[]”合并相同域的数据并排成水平向量：

```
 all_x=[ps.name]

all_x =
曲线 1 曲线 2
```

- 使用 cat 将其变成多维数组：

```
 cat(1,ps.position)       %沿第一维排列

ans =
```

```
     0     0   300   300
   100   100   300   300
 cat(2,ps.position)      %沿第二维排列

ans =
     0     0   300   300   100   100   300   300
 cat(3,ps.position)      %沿第三维排列

ans(:,:,1) =
     0     0   300   300
ans(:,:,2) =
   100   100   300   300
```

2.6 数据分析

数据分析按照以下原则：

- 如果输入是向量，则按整个向量进行运算。
- 如果输入是矩阵，则按列进行运算。

2.6.1 数据统计和相关分析

相关分析包括计算协方差和相关系数，相关系数越大说明相关性越强。

例如，用某年一月份中连续四天的温度数据构成 4×3 的矩阵 a，包括最高温度、最低温度和平均温度如表 2.9 所示。列为按最高温度、最低温度和平均温度进行分类，而行是每天的温度数据样本。

表 2.9 四天的温度

平均温度	最高温度	最低温度
5.30	13.00	0.40
5.10	11.80	-1.70
3.70	8.10	0.60
1.50	7.70	-4.50

对该矩阵进行简单数据统计分析，MATLAB 函数如表 2.10 所示。

表 2.10 数据统计分析函数

函数名	功能	例子结果		
max(X)	矩阵中各列的最大值。	5.3000	13.0000	0.6000
min(X)	矩阵中各列最小值。	1.5000	7.7000	-4.5000
mean(X)	矩阵中各列平均值。	3.9000	10.1500	-1.3000
std(X)	矩阵中各列标准差，指各元素与该列平均值(mean)之差的平方和开方。	3.9000	10.1500	-1.3000

median(X)	矩阵中各列的中间元素。	4.4000	9.9500	-0.6500
var(X)	矩阵中各列的方差。	3.0667	7.0167	5.6333
C=cov(X)	矩阵中各列间的协方差。	3.0667 4.0867 3.0667	4.0867 7.0167 2.7100	3.0667 2.7100 5.6333
S=corrcoef(X)	矩阵中各列间的相关系数矩阵，与协方差 C 的关系为：S(i,j)=C(i,j)/$\sqrt{C(i,i)C(j,j)}$。对角线为 x 和 y 的自相关系数。	1.0000 0.8810 0.7378	0.8810 1.0000 0.4310	0.7378 0.4310 1.0000
[S,k]=sort(X,n)	沿第 n 维按模增大重新排序，k 为 S 元素的原位置。	sort(a,1)= 1.5000 3.7000 5.1000 5.3000	 7.7000 8.1000 11.8000 13.0000	 -4.5000 -1.7000 0.4000 0.6000

2.6.2 差分和积分

MATLAB 可以通过函数对矩阵的差分和积分等进行方便地运算。常用的差分和积分函数如表 2.11 所示。

其中矩阵 a 仍然是前例中的温度数据：

```
a =
    5.3000   13.0000    0.4000
    5.1000   11.8000   -1.7000
    3.7000    8.1000    0.6000
    1.5000    7.7000   -4.5000
```

表 2.11　差分和累计的数据处理函数

函数名	功能	例子			
		输入	结果		
diff(X,m,n)	沿第 n 维求第 m 阶列向差分。差分是求相邻行之间的差，结果会减少一行。	diff(a,1,1) %沿第一维求一阶差分	-0.2000 -1.4000 -2.2000	-1.2000 -3.7000 -0.4000	-2.1000 2.3000 -5.1000
[fx,fy]=gradient(Z)	对 Z 求 x、y 方向的数值梯度。	gradient(a) %对列求数值梯度	7.7000 6.7000 4.4000 6.2000	-2.4500 -3.4000 -1.5500 -3.0000	-12.6000 -13.5000 -7.5000 -12.2000
sum(X)	矩阵各列元素的和。	sum(a)	15.6000	40.6000	-5.2000
cumsum(X,n)	沿第 n 维求累计和	cumsum(a,2) %沿列求累计和	5.3000 5.1000 3.7000 1.5000	18.3000 16.9000 11.8000 9.2000	18.7000 15.2000 12.4000 4.7000

cumprod(X,n)	沿第 n 维求累计乘积	cumprod(a,2) %沿列求累计乘积	5.3000 68.9000 27.5600 5.1000 60.1800 -102.3060 3.7000 29.9700 17.9820 1.5000 11.5500 -51.9750
trapz(X,y)	梯形法求积分近似于求元素和，把相邻两点数据的平均值乘以步长表示面积。x 为自变量，y 为函数。	trapz(a)	12.2000 30.2500 -3.1500
cumtrapz(X,y,n)	用梯形法沿第 n 维求函数 y 对自变量 x 累计积分。	cumtrapz(a)%用梯形法沿列向积分	0 0 0 5.2000 12.4000 -0.6500 9.6000 22.3500 -1.2000 12.2000 30.2500 -3.1500

【例 2.26】已知 $y = e^{-0.2}\sin(t)$，其中 t 的范围是[0 10]，计算 y 的微分和积分。y 的微分和积分曲线如图 2.9 所示。

```
t=0:0.5:10;
y=exp(-0.2).*sin(t)

y =
  Columns 1 through 7
        0    0.3925    0.6889    0.8167    0.7445    0.4900    0.1155
  Columns 8 through 14
   -0.2872    -0.6196    -0.8003    -0.7851    -0.5776    -0.2288
0.1761
  Columns 15 through 21
    0.5379      0.7680      0.8100      0.6537      0.3374     -0.0615
-0.4454
d=[0 diff(y)]           %计算微分

d =
  Columns 1 through 7
        0    0.3925    0.2964    0.1277   -0.0722   -0.2545   -0.3744
  Columns 8 through 14
   -0.4027    -0.3324    -0.1807      0.0152      0.2075      0.3489
0.4049
  Columns 15 through 21
    0.3618      0.2301      0.0420     -0.1563     -0.3163     -0.3989
-0.3839
s1=0.5*cumsum(y)        %用矩形法计算积分，横坐标两点间隔为 0.5

s1 =
  Columns 1 through 7
        0    0.1963    0.5407    0.9491    1.3213    1.5663    1.6241
```

```
  Columns 8 through 14
    1.4805    1.1707    0.7705    0.3779    0.0891   -0.0253    0.0628
  Columns 15 through 21
    0.3317    0.7157    1.1207    1.4476    1.6163    1.5856    1.3629
s2=cumtrapz(t,y)           %用梯形法计算积分

s2 =
  Columns 1 through 7
         0    0.0981    0.3685    0.7449    1.1352    1.4438    1.5952
  Columns 8 through 14
    1.5523    1.3256    0.9706    0.5742    0.2335    0.0319    0.0188
  Columns 15 through 21
    0.1973    0.5237    0.9182    1.2842    1.5320    1.6009    1.4742
```

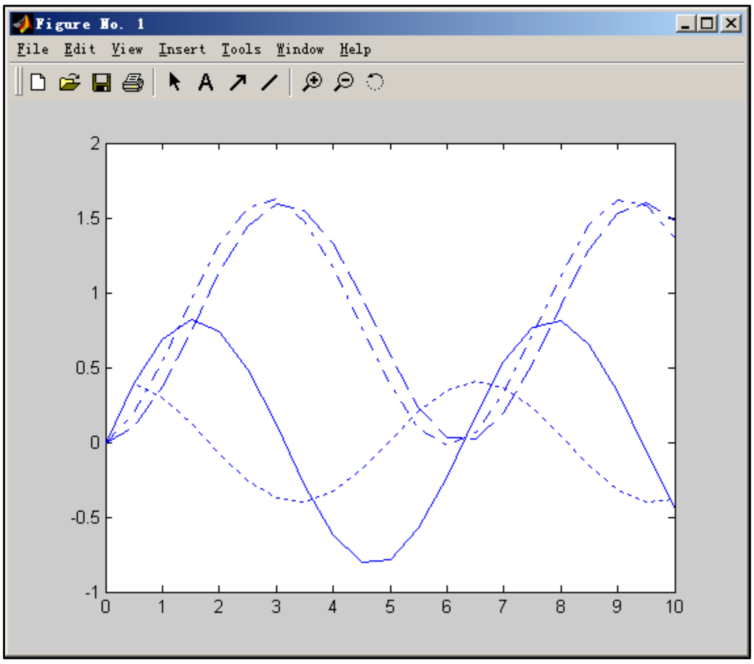


图 2.9 微分和积分曲线图

2.6.3 卷积和快速傅里叶变换

1. 卷积

卷积和解卷是信号与系统中常用的数学工具。函数 conv 和 deconv 分别为卷积和解卷函数，同时也是多项式乘法和除法(2.4.2 小节)函数。

- conv：计算向量的卷积。

语法：

conv(x,y)

如果 x 是输入信号，y 是线性系统的脉冲过渡函数，则 x 和 y 的卷积为系统的输出信号。

- conv2：计算二维卷积。
- deconv：解卷积运算。

语法：

[q,r]=deconv(x,y)

解卷积和卷积的关系是：x=conv(y,q)+r。

2. 快速傅立叶变换

- fft：一维快速傅立叶变换。

语法：

X=fft(x,N)　%对离散序列进行离散傅立叶变换

说明：x 可以是向量、矩阵和多维数组；N 为输入变量 x 的序列长度，可省略，如果 X 的长度小于 N，则会自动补零；如果 X 的长度大于 N，则会自动截断；当 N 取 2 的整数幂时，傅立叶变换的计算速度最快。通常取大于又最靠近 x 长度的幂次。

一般情况下，fft 求出的函数为复数，可用 abs 及 angle 分别求其幅值和相位。

- ifft：一维快速傅立叶逆变换。

语法：

X=ifft(x,N)　%对离散序列进行离散傅立叶逆变换

【例 2.27】利用傅立叶变换和卷积公式求两个离散序列的卷积。

已知：$A(n)=\begin{cases}0 & 其它\\ 1 & n=2,3,\cdots 10\end{cases}$　$B(n)=\begin{cases}0 & 其它\\ 1 & n=4,5,\cdots 9\end{cases}$

```
A=ones(1,10);
A(1)=0

A =
     0     1     1     1     1     1     1     1     1     1
B=ones(1,9);
B([1 2 3])=0

B =
     0     0     0     1     1     1     1     1     1
C=conv(A,B)            %计算卷积

C =
  Columns 1 through 13
     0     0     0     0     1     2     3     4     5     6     6     6
6
  Columns 14 through 18
     5     4     3     2     1
N=32;                  %序列长度为 32
AF=fft(A,N);               %傅立叶变换
BF=fft(B,N);
CF=AF.*BF;
CC=real(ifft(CF));         %过滤掉虚部
```

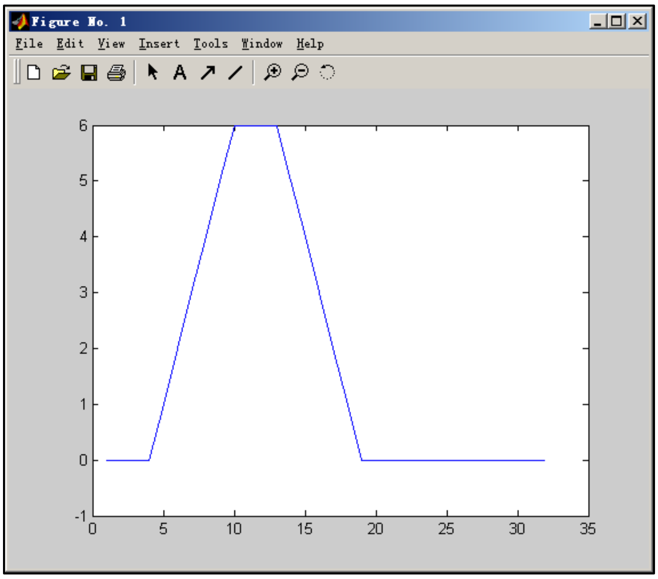


图 2.10 卷积结果

程序分析：可以看到直接计算的卷积结果 C 和用傅立叶变换求卷积的 CC 结果相同，如图 2.10 所示。其中序列长度 N 的取值应为 2 的整数幂，必须不小于 length(A)+length(B)-1。

2.6.4 向量函数

- 两个向量的矢量积(叉乘)

语法：

cross(a,b)

- 两个向量的数量积(点乘)

语法：

dot(a,b)

说明：通常 a、b 为包含 3 个元素的向量。

【例 2.28】计算向量的叉乘和点乘。

```
a=[1 2 3];
b=[4 5 6];
c=cross(a,b)

c =
    -3     6    -3
c=dot(a,b)

c =
    32
```

第 3 章　MATLAB 符号计算

符号计算则是可以对未赋值的符号对象(可以是常数、变量、表达式)进行运算和处理。MATLAB 具有符号数学工具箱(Symbolic Math Toolbox)，将符号运算结合到 MATLAB 的数值运算环境。符号数学工具箱是建立在 Maple 软件基础上的。

3.1 符号表达式的建立

Symbolic Math Toolbox2.1 版规定在进行符号计算时，首先要定义基本的符号对象然后才能进行符号运算。

3.1.1 创建符号常量

符号常量是不含变量的符号表达式，用 sym 命令来创建符号常量。

语法：

sym(‘常量’)　　　　%创建符号常量

例如，创建符号常量，这种方式是绝对准确的符号数值表示：

```
>> a=sym('sin(2)')
a =
sin(2)
```

sym 命令也可以把数值转换成某种格式的符号常量。

语法：

sym(常量,参数)　　　%把常量按某种格式转换为符号常量

说明：参数可以选择为’d’、’f’、’e’或’r’ 四种格式，也可省略，其作用如表 3.1 所示。

表 3.1 参数设置

参数	作用
d	返回最接近的十进制数值(默认位数为 32 位)
f	返回该符号值最接近的浮点表示
r	返回该符号值最接近的有理数型(为系统默认方式)，可表示为 p/q、p*q、10^q、pi/q、2^q 和 sqrt(p)形式之一
e	返回最接近的带有机器浮点误差的有理值

例如，创建符号常量，这种方式是绝对准确的符号数值表示：

```
a=sym('sin(2)')

a =
sin(2)
```

例如，把常量转换为符号常量，按系统默认格式转换：

```
a=sym(sin(2))

a =
8190223105242182*2^(-53)
```

【例 3.1】创建数值常量和符号常量。

```
 a1=2*sqrt(5)+pi                %创建数值常量

a1 =
   7.6137
 a2=sym('2*sqrt(5)+pi')          %创建符号表达式

a2 =
2*sqrt(5)+pi
 a3=sym(2*sqrt(5)+pi)           %按最接近的有理数型表示符号常量

a3 =
8572296331135796*2^(-50)
 a4=sym(2*sqrt(5)+pi,'d')        %按最接近的十进制浮点数表示符号常量

a4 =
7.6137286085893727261009189533070
 a31=a3-a1                      %数值常量和符号常量的计算

a31 =
0
 a5='2*sqrt(5)+pi'              %字符串常量

a5 =
2*sqrt(5)+pi
```

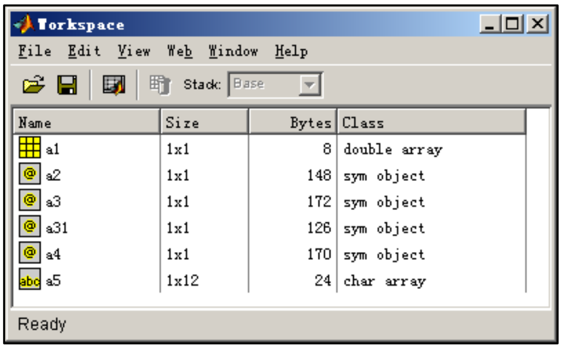


图 3.1 工作空间窗口

可以通过查看工作空间来查看各变量的数据类型和存储空间，工作空间如图 3.1 所示。

3.1.2 创建符号变量和表达式

创建符号变量和符号表达式可以使用 sym 和 syms 命令。

1. 使用 sym 命令创建符号变量和表达式

语法：

sym('变量',参数)　　　　%把变量定义为符号对象

说明：参数用来设置限定符号变量的数学特性，可以选择为'positive'、'real'和'unreal'，'positive'表示为"正、实"符号变量，'real'表示为"实"符号变量，'unreal'表示为"非实"符号变量。如果不限定则参数可省略。

【例 3.2】创建符号变量，用参数设置其特性。

```
syms x y real                %创建实数符号变量
z=x+i*y;                     %创建 z 为复数符号变量
real(z)                      %复数 z 的实部是实数 x

ans =
x
sym('x','unreal');           %清除符号变量的实数特性
real(z)                      %复数 z 的实部

ans =
1/2*x+1/2*conj(x)
```

程序分析：设置 x、y 为实数型变量，可以确定 z 的实部和虚部。

语法：

sym('表达式')　　　　%创建符号表达式

【例 3.2 续】创建符号表达式。

```
f1=sym('a*x^2+b*x+c')

f1 =
a*x^2+b*x+c
```

2.使用 syms 命令创建符号变量和符号表达式

语法：

syms('arg1', 'arg2', …,参数)　　　%把字符变量定义为符号变量

syms arg1 arg2 …,参数　　　%把字符变量定义为符号变量的简洁形式

说明：syms 用来创建多个符号变量，这两种方式创建的符号对象是相同的。参数设置和前面的 sym 命令相同，省略时符号表达式直接由各符号变量组成。

【例 3.2 续】使用 syms 命令创建符号变量和符号表达式。

```
syms a b c x                 %创建多个符号变量
f2=a*x^2+b*x+c               %创建符号表达式

f2 =
a*x^2+b*x+c
syms('a','b','c','x')
f3=a*x^2+b*x+c;              %创建符号表达式
```

程序分析：既创建了符号变量 a、b、c、x，又创建了符号表达式，f2、f3 和 f1 符号表达式相同。

3.1.3 符号矩阵

用 sym 和 syms 命令也可以创建符号矩阵。

```
A=sym('[a,b;c,d]')

A =
[ a, b]
[ c, d]
```
例如，使用 syms 命令创建相同的符号矩阵：
```
syms a b c d
A=[a b;c d]

A =
[ a, b]
[ c, d]
```
【例 3.3】比较符号矩阵与字符串矩阵的不同。
```
A=sym('[a,b;c,d]')       %创建符号矩阵

A =
[ a, b]
[ c, d]

B='[a,b;c,d]'            %创建字符串矩阵

B =
[a,b;c,d]

C=[a,b;c,d]          %创建数值矩阵

??? Undefined function or variable 'a'.
```
程序分析：由于数值变量 a、b、c、d 未事先赋值，MATLAB 给出错误信息。
```
C=sym(B)                 %转换为符号矩阵

C =
[ a, b]
[ c, d]
whos

  Name      Size                Bytes  Class

  A         2x2                  312  sym object
  B         1x9                   18  char array
  C         2x2                  312  sym object

Grand total is 25 elements using 642 bytes
```
程序分析：查看符号矩阵 A，可以看到为 2×2 的符号矩阵，占用较多的字节。

3.2 符号表达式的代数运算

符号运算与数值运算的区别主要有以下几点：

- 传统的数值型运算因为要受到计算机所保留的有效位数的限制，它的内部表示法总是采用计算机硬件提供的 8 位浮点表示法，因此每一次运算都会有一定的截断误差，重复的多次数值运算就可能会造成很大的累积误差。符号运算不需要进行数值运算，不会出现截断误差，因此符号运算是非常准确的。
- 符号运算可以得出完全的封闭解或任意精度的数值解。
- 符号运算的时间较长，而数值型运算速度快。

3.2.1 符号表达式的代数运算

符号表达式的运算符和基本函数都与数值计算中的几乎完全相同。

1. 符号运算中的运算符

(1) 基本运算符

- 运算符“＋”，“－”，“*”，“\”，“/”，“^”分别实现符号矩阵的加、减、乘、左除、右除、求幂运算。
- 运算符“.*”，“./”，“.\”，“.^”分别实现符号数组的乘、除、求幂，即数组间元素与元素的运算。
- 运算符“' ”，“.' ”分别实现符号矩阵的共轭转置、非共轭转置。

(2) 关系运算符

- 在符号对象的比较中，没有“大于”、“大于等于”、“小于”、“小于等于”的概念，而只有是否“等于”的概念。
- 运算符“= =”、“~=”分别对运算符两边的符号对象进行“相等”、“不等”的比较。当为“真”时，比较结果用 1 表示；当为“假”时，比较结果则用 0 表示。

2. 函数运算

(1) 三角函数和双曲函数

三角函数包括 sin、cos、tan；双曲函数包括 sinh、cosh、tanh；三角反函数除了 atan2 函数仅能用于数值计算外，其余的 asin、acos、atan 函数在符号运算中与数值计算的使用方法相同。

(2) 指数和对数函数

指数函数 sqrt、exp、expm 的使用方法与数值计算的完全相同；对数函数在符号计算中只有自然对数 log(表示 ln)，而没有数值计算中的 log2 和 log10。

(3) 复数函数

复数的共轭 conj、求实部 real、求虚部 imag 和求模 abs 函数与数值计算中的使用方法相同。但注意，在符号计算中，MATLAB 没有提供求相角的命令。

(4) 矩阵代数命令

MATLAB 提供的常用矩阵代数命令有 diag，triu，tril，inv，det，rank， poly，expm，eig 等，它们的用法几乎与数值计算中的情况完全一样。

【例 3.4】求矩阵 $A=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ 的行列式值、非共轭转置和特征值。

```
syms a11 a12 a21 a22
A=[a11 a12;a21 a22]             %创建符号矩阵

A =
[ a11, a12]
[ a21, a22]
det(A)                     %计算行列式

ans =
a11*a22-a12*a21
A.'                        %计算非共轭转置

ans =
[ a11, a21]
[ a12, a22]
eig(A)                     %计算特征值

ans =
[ 1/2*a11+1/2*a22+1/2*(a11^2-2*a11*a22+a22^2+4*a12*a21)^(1/2)]
[ 1/2*a11+1/2*a22-1/2*(a11^2-2*a11*a22+a22^2+4*a12*a21)^(1/2)]
```

【例 3.5】符号表达式 $f=2x^2+3x+4$ 与 $g=5x+6$ 的代数运算。

```
f=sym('2*x^2+3*x+4')

f =
2*x^2+3*x+4
g=sym('5*x+6')

g =
5*x+6
f+g                     %符号表达式相加

ans =
2*x^2+8*x+10
f*g                     %符号表达式相乘

ans =
(2*x^2+3*x+4)*(5*x+6)
```

3.2.2 符号数值任意精度控制和运算

1. Symbolic Math Toolbox 中的算术运算方式

在 Symbolic Math Toolbox 中有三种不同的算术运算：

- 数值型：MATLAB 的浮点运算。
- 有理数型：Maple 的精确符号运算。
- VPA 型：Maple 的任意精度运算。

2. 任意精度控制

任意精度的 VPA 型运算可以使用 digits 和 vpa 命令来实现。

语法：

digits(n)　　　　　%设定默认的精度

说明：n 为所期望的有效位数。digits 函数可以改变默认的有效位数来改变精度，随后的每个进行 Maple 函数的计算都以新精度为准。当有效位数增加时，计算时间和占用的内存也增加。命令“digits”用来显示默认的有效位数，默认为 32 位。

语法：

S=vpa(s,n)　　　　　%将 s 表示为 n 位有效位数的符号对象

说明：s 可以是数值对象或符号对象，但计算的结果 S 一定是符号对象；当参数 n 省略时则以给定的 digits 指定精度。vpa 命令只对指定的符号对象 s 按新精度进行计算，并以同样的精度显示计算结果，但并不改变全局的 digits 参数。

【例 3.6】对表达式 $2\sqrt{5}+\pi$ 进行任意精度控制的比较。

```
a=sym('2*sqrt(5)+pi')

a =
2*sqrt(5)+pi
digits          %显示默认的有效位数


Digits = 32

vpa(a)          %用默认的位数计算并显示

ans =
7.6137286085893726312809907207421
vpa(a,20)          %按指定的精度计算并显示

ans =
7.6137286085893726313
digits(15)          %改变默认的有效位数
vpa(a)          %按 digits 指定的精度计算并显示

ans =
7.61372860858937
```

3. Symbolic Math Toolbox 中的三种运算方式的比较

【例 **3.6** 续】用三种运算方式表达式比较 2/3 的结果。

```
a1 =2/3        %数值型

a1 =
   0.6667
a2 = sym(2/3)     %有理数型

a2 =
2/3
 a3 =vpa('2/3',32)   %VPA 型

a3 =
.66666666666666666666666666666667
```

程序分析：

- 三种运算方式中数值型运算的速度最快。
- 有理数型符号运算的计算时间和占用内存是最大的，产生的结果是非常准确的。
- VPA 型的任意精度符号运算比较灵活，可以设置任意有效精度，当保留的有效位数增加时，每次运算的时间和使用的内存也会增加。
- 数值型变量 a1 结果显示的有效位数并不是存储的有效位数，在第一章中介绍显示的有效位数由“format”命令控制。如下面修改“format”命令就改变了显示的有效位数：

```
 format long
 a1

a1 =
  0.66666666666667
```

3.2.3 符号对象与数值对象的转换

1. 将数值对象转换为符号对象

sym 命令可以把数值型对象转换成有理数型符号对象，vpa 命令可以将数值型对象转换为任意精度的 VPA 型符号对象。

2. 将符号对象转换为数值对象

使用 double、 numeric 函数可以将有理数型和 VPA 型符号对象转换成数值对象。

语法：

N=double(S)　　　%将符号变量 S 转换为数值变量 N

N=numeric(S)　　　%将符号变量 S 转换为数值变量 N

【例 **3.7**】将符号变量 $2\sqrt{5}+\pi$ 与数值变量进行转换。

```
 clear
a1=sym('2*sqrt(5)+pi')
```

```
a1 =
2*sqrt(5)+pi

 b1=double(a1)                  %转换为数值变量

b1 =
    7.6137

 a2=vpa(sym('2*sqrt(5)+pi'),32)

a2 =
7.6137286085893726312809907207421

 b2=numeric(a2)             %转换为数值变量

b2 =
    7.6137
```

【例 3.7 续】由符号变量得出数值结果。

```
 b3=eval(a1)

b3 =
    7.6137
```

用“whos”命令查看变量的类型，可以看到 b1、b2、b3 都转换为双精度型：

```
 whos

  Name      Size                   Bytes  Class

  a1        1x1                      148  sym object
  a2        1x1                      190  sym object
  b1        1x1                        8  double array
  b2        1x1                        8  double array
  b3        1x1                        8  double array

Grand total is 50 elements using 362 bytes
```

3.3 符号表达式的操作和转换

3.3.1 符号表达式中自由变量的确定

1. 自由变量的确定原则

，MATLAB 将基于以下原则选择一个自由变量：

- 小写字母 i 和 j 不能作为自由变量。
- 符号表达式中如果有多个字符变量，则按照以下顺序选择自由变量：首先选择 x 作

为自由变量；如果没有 x，则选择在字母顺序中最接近 x 的字符变量；如果与 x 相同距离，则在 x 后面的优先。

- 大写字母比所有的小写字母都靠后。

2. findsym 函数

如果不确定符号表达式中的自由符号变量，可以用 findsym 函数来自动确定。

语法：

findsym(EXPR,n)　　%确定自由符号变量

说明：EXPR 可以是符号表达式或符号矩阵；n 为按顺序得出符号变量的个数，当 n 省略时，则不按顺序得出 EXPR 中所有的符号变量。

【**例 3.8**】得出符号表达式中的符号变量。

```
 f=sym('a*x^2+b*x+c')

f =
a*x^2+b*x+c
 findsym(f)              %得出所有的符号变量

ans =
a, b, c, x
 g=sym('sin(z)+cos(v)')

g =
sin(z)+cos(v)
 findsym(g,1)            %得出第一个符号变量

ans =
z
```

程序说明：符号变量 z 和 v 距离 x 相同，以在 x 后面的 z 为自由符号变量。

3.3.2 符号表达式的化简

同一个数学函数的符号表达式的可以表示成三种形式，例如以下的 f(x)就可以分别表示为：

- 多项式形式的表达方式：$f(x)=x^3+6x^2+11x-6$
- 因式形式的表达方式：$f(x)=(x-1)(x-2)(x-3)$
- 嵌套形式的表达方式：$f(x)=x(x(x-6)+11)-6$

【**例 3.9**】三种形式的符号表达式的表示。

```
 f=sym('x^3-6*x^2+11*x-6')           %多项式形式

f =
x^3-6*x^2+11*x-6
 g= sym('(x-1)*(x-2)*(x-3)')           %因式形式

g =
```

```
(x-1)*(x-2)*(x-3)
h= sym(' x*(x*(x-6)+11)-6')            %嵌套形式

h =
x*(x*(x-6)+11)-6
```

1. pretty 函数

【例 3.9 续】给出相应的符号表达式形式。

```
pretty(f)


                        3      2
                       x  - 6 x  + 11 x - 6
```

2. collect 函数

【例 3.9 续】给出相应的符号表达式形式。

```
collect(g)

ans =
x^3-6*x^2+11*x-6
```

当有多个符号变量，可以指定按某个符号变量来合并同类项。下面有 x、y 符号变量的表达式：

```
f1=sym('x^3+2*x^2*y+4*x*y+6')

f1 =
x^3+2*x^2*y+4*x*y+6
collect(f1,'y')              %按 y 来合并同类项

ans =
(2*x^2+4*x)*y+x^3+6
```

3. expand 函数

【例 3.9 续】给出相应的符号表达式形式。

```
expand(g)

ans =
x^3-6*x^2+11*x-6
```

4. horner 函数

【例 3.9 续】给出符号表达式的嵌套形式。

```
horner(f)

ans =
x*(x*(x-6)+11)-6
```

5. factor 函数

【例 3.9 续】给出符号表达式的因式形式。

```
factor(f)

ans =
(x-1)*(x-2)*(x-3)
```

6. simplify 函数

【例 3.9 续】利用三角函数来简化符号表达式 $\cos^2x-\sin^2x$。

```
y=sym('cos(x)^2-sin(x)^2')

y =
cos(x)^2-sin(x)^2
simplify(y)

ans =
2*cos(x)^2-1
```

7. simple 函数

simple 函数给出多种简化形式，给出除了 pretty 、collect、expand、factor、simplify 简化形式之外的 radsimp、combine、combine(trig)、convert 形式，并寻求包含最少数目字符的表达式简化形式。

【例 3.9 续】利用 simple 简化符号表达式 $\cos^2x-\sin^2x$。

```
simple(y)


simplify:

2*cos(x)^2-1

radsimp:

cos(x)^2-sin(x)^2

combine(trig):

cos(2*x)

factor:

(cos(x)-sin(x))*(cos(x)+sin(x))

expand:
```

```
cos(x)^2-sin(x)^2

combine:

cos(2*x)

convert(exp):

(1/2*exp(i*x)+1/2/exp(i*x))^2+1/4*(exp(i*x)-1/exp(i*x))^2

convert(sincos):

cos(x)^2-sin(x)^2

convert(tan):

(1-tan(1/2*x)^2)^2/(1+tan(1/2*x)^2)^2-4*tan(1/2*x)^2/(1+tan(1/2*x)^2)^2

collect(x):

cos(x)^2-sin(x)^2
ans =
cos(2*x)
```

程序分析：得出最简化的符号表达式为“cos(2*x)”。

3.3.3 符号表达式的替换

1. subexpr 函数

语法：

subexpr(s,s1)　　　　%用符号变量 s1 来置换 s 中的子表达式

subexpr 函数对子表达式是自动寻找的，只有比较长的子表达式才被置换，比较短的子表达式，即使重复出现多次，也不被置换。

【**例 3.10**】用 subexpr 函数使 $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 的特征值表达式简洁。

```
syms a b c d x
s=eig([a b;c d])            %计算特征值

s =
[ 1/2*a+1/2*d+1/2*(a^2-2*a*d+d^2+4*b*c)^(1/2)]
[ 1/2*a+1/2*d-1/2*(a^2-2*a*d+d^2+4*b*c)^(1/2)]
subexpr(s,x)                %用 x 替换子表达式
```

```
ans =
[ 1/2*a+1/2*d+1/2*(a^2-2*a*d+d^2+4*b*c)^(1/2)]
[ 1/2*a+1/2*d-1/2*(a^2-2*a*d+d^2+4*b*c)^(1/2)]
```

2. subs 函数

subs 函数可用来进行对符号表达式中符号变量的替换。

语法：

subs(s) %用给定值替换符号表达式 s 中的所有变量

subs(s,new) %用 new 替换符号表达式 s 中的自由变量

subs(s,old,new) %用 new 替换符号表达式 s 中的 old 变量

【**例 3.10 续**】用 subs 函数对符号表达式$(x+y)^2+3(x+y)+5$ 进行替换。

```
f=sym('(x+y)^2+3*(x+y)+5')      %创建符号表达式

f =
(x+y)^2+3*(x+y)+5

x=5;
f1=subs(f)                     %用工作空间的给定值替换 x

f1 =
(5+y)^2+20+3*y
f2=subs(f,'x+y','s')              %用 s 替换 x+y

f2 =
((s))^2+3*((s))+5
f3=subs(f,'x+y',5)             %用常数 5 替换 x+y

f3 =
   45
f4=subs(f,'x','z')             %用 z 替换 x

f4 =
((z)+y)^2+3*((z)+y)+5
```

3.3.4 求反函数和复合函数

在 MATLAB 中 finverse 函数可以求得符号函数的反函数。

语法：

finverse(f,v) %对指定自变量 v 的函数 f(v)求反函数

说明：当 v 省略，则对默认的自由符号变量求反函数。

1. 求反函数

【**例 3.11**】求 te^x 的反函数。

```
f=sym('t*e^x')             %原函数
```

```
f =
t*e^x
 g=finverse(f)              %对默认自由变量求反函数

g =
log(x/t)/log(e)
 g=finverse(f,'t')              %对 t 求反函数

g =
t/(e^x)
```

程序分析：如果先定义 t 为符号变量，则参数't'的单引号可去掉：

```
 syms t
 g=finverse(f,t)
```

2. 求复合函数

【例 3.11 续】计算 te^x 与 ay^2+by+c 的复合函数。

```
 f=sym('t*e^x');                %创建符号表达式
 g=sym('a*y^2+b*y+c');          %创建符号表达式
 h1=compose(f,g)            %计算 f(g(x))

h1 =
t*e^(a*y^2+b*y+c)
 h2=compose(g,f)            %计算 g(f(x))

h2 =
a*t^2*(e^x)^2+b*t*e^x+c
 h3=compose(f,g,'z')            %计算 f(g(z))

h3 =
t*e^(a*z^2+b*z+c)
```

【例 3.11 续】计算得出 te^x 与 y^2 的复合函数。

```
 f1=sym('t*e^x');
 g1=sym('y^2');
 h1=compose(f1,g1)

h1 =
t*e^(y^2)
 h2=compose(f1,g1,'z')   %计算 f(g(z))

h2 =
t*e^(z^2)
 h3=compose(f1,g1,'t','y')       %以 t 为自变量计算 f(g(z))

h3 =
```

```
y^2*e^x
 h4=compose(f1,g1,'t','y','z')  %以 t 为自变量计算 f(g(z))，并用 z 替换 y

h4 =
z^2*e^x
 h5=subs(h3,'y','z')          %用替换的方法实现 h5 与 h4 相同结果

h5 =
(z)^2*e^x
```

3.3.5 符号表达式的转换

1. 符号表达式与多项式的转换

构成多项式的符号表达式 f(x)可以与多项式系数构成的行向量进行相互转换，MATLAB 提供了函数 sym2poly 和 poly2sym 实现相互转换。

(1) sym2poly 函数

【例 3.12】将符号表达式 $2x+3x^2+1$ 转换为行向量。

```
 f=sym('2*x+3*x^2+1')

f =
2*x+3*x^2+1
 sym2poly(f)          %转换为按降幂排列的行向量

ans =
    3     2     1
 f1=sym('a*x^2+b*x+c')

f1 =
a*x^2+b*x+c
 sym2poly(f1)

??? Error using ==> sym/sym2poly
Input has more than one symbolic variable.
```

程序分析：只能对含有一个变量的符号表达式进行转换。

(2) poly2sym 函数

【例 3.12 续】将行向量转换为符号表达式。

```
 g=poly2sym([1 3 2])         %默认 x 为符号变量的符号表达式

g =
x^2+3*x+2
 g=poly2sym([1 3 2],sym('y')) %y 为符号变量的符号表达式

g =
y^2+3*y+2
```

2. 提取分子和分母

如果符号表达式是一个有理分式(两个多项式之比)，可以利用 numden 函数来提取分子或分母，还可以进行通分。

语法：

[n,d]=numden(f)

说明：n 为分子；d 为分母；f 为有理分式。

【例 3.13】用 numden 函数来提取符号表达式 $\frac{1}{s^2+3s+2}$ 和 $\frac{1}{s^2}+3s+2$ 的分子、分母。

```
 f1=sym('1/(s^2+3*s+2)')

f1 =
1/(s^2+3*s+2)
 f2=sym('1/s^2+3*s+2')

f2 =
1/s^2+3*s+2
 [n1,d1]=numden(f1)

n1 =
1
d1 =
s^2+3*s+2
 [n2,d2]=numden(f2)

n2 =
1+3*s^3+2*s^2
d2 =
s^2
```

3.4 符号极限、微积分和级数求和

3.4.1 符号极限

假定符号表达式的极限存在，Symbolic Math Toolbox 提供了直接求表达式极限的函数 limit，函数 limit 的基本用法如表 3.2 所示。

表 3.2 limit 函数的用法

表达式	函数格式	说明
$\lim_{x \to 0} f(x)$	limt(f)	对 x 求趋近于 0 的极限
$\lim_{x \to a} f(x)$	limt(f,x,a)	对 x 求趋近于 a 的极限，当左右极限不相等时极限不存在。

$\lim_{x \to a^-} f(x)$	limt(f,x,a, left)	对 x 求左趋近于 a 的极限
$\lim_{x \to a^+} f(x)$	limt(f,x,a, right)	对 x 求右趋近于 a 的极限

【例 3.14】分别求 1/x 在 0 处从两边趋近、从左边趋近和从右边趋近的三个极限值。

```
 f=sym('1/x')

f =
1/x
 limit(f)              %对 x 求趋近于 0 的极限

ans =
NaN
 limit(f,'x',0)        %对 x 求趋近于 0 的极限

ans =
NaN
 limit(f,'x',0,'left')   %左趋近于 0

ans =
-inf
 limit(f,'x',0,'right')     %右趋近于 0

ans =
inf
```

程序分析：当左右极限不相等，表达式的极限不存在为 NaN。

采用极限方法也可以用来求函数的导数：$f'(x) = \lim_{t \to 0} \frac{f(x+t) - f(x)}{t}$。

【例 3.14 续】求函数 cos(x)的导数。

```
 syms t x
 limit((cos(x+t)-cos(x))/t,t,0)

ans =
-sin(x)
```

3.4.2 符号微分

函数 diff 是用来求符号表达式的微分。

语法：

diff(f)　　%求 f 对自由变量的一阶微分

diff(f,t)　　%求 f 对符号变量 t 的一阶微分

diff(f,n)　　%求 f 对自由变量的 n 阶微分

diff(f,t,n)　　%求 f 对符号变量 t 的 n 阶微分

【例 3.15】已知 f(x)＝ax^2+bx+c，求 f(x)的微分。

```
f=sym('a*x^2+b*x+c')

f =
a*x^2+b*x+c
diff(f)   %对默认自由变量 x 求一阶微分

ans =
2*a*x+b
diff(f,'a')   %对符号变量 a 求一阶微分

ans =
x^2
diff(f,'x',2)   %对符号变量 x 求二阶微分

ans =
2*a
diff(f,3)     %对默认自由变量 x 求三阶微分

ans =
0
```

微分函数 diff 也可以用于符号矩阵，其结果是对矩阵的每一个元素进行微分运算。

【例 3.15 续】对符号矩阵$\begin{bmatrix} 2x & t^2 \\ t\sin(x) & e^x \end{bmatrix}$求微分。

```
syms t x
g=[2*x t^2;t*sin(x) exp(x)]  %创建符号矩阵

g =
[      2*x,      t^2]
[ t*sin(x),   exp(x)]
diff(g)                  %对默认自由变量 x 求一阶微分

ans =
[        2,        0]
[ t*cos(x),   exp(x)]
diff(g,'t')                 %对符号变量 t 求一阶微分

ans =
[      0,    2*t]
[ sin(x),      0]
diff(g,2)                   %对默认自由变量 x 求二阶微分
```

```
ans =
[        0,        0]
[ -t*sin(x),    exp(x)]
```

diff 还可以用于对数组中的元素进行逐项求差值。

【例 3.15 续】可以使用 diff 计算向量间元素的差值。

```
x1=0:0.5:2;
y1=sin(x1)

y1 =
       0    0.4794    0.8415    0.9975    0.9093
diff(y1)                       %计算元素差

ans =
   0.4794    0.3620    0.1560   -0.0882
```

程序分析：计算出的差值比原来的向量少一列。

3.4.3 符号积分

积分有定积分和不定积分，运用函数 int 可以求得符号表达式的积分。

语法：

int(f,'t') %求符号变量 t 的不定积分

int(f,'t',a,b) %求符号变量 t 的积分

int(f,'t','m','n') %求符号变量 t 的积分

说明：t 为符号变量，当 t 省略则为默认自由变量；a 和 b 为数值，[a,b]为积分区间；m 和 n 为符号对象，[m,n]为积分区间；与符号微分相比，符号积分复杂得多。因为函数的积分有时可能不存在，即使存在，也可能限于很多条件，MATLAB 无法顺利得出。当 MATLAB 不能找到积分时，它将给出警告提示并返回该函数的原表达式。

【例 3.16】求积分 $\int \cos(x)$ 和 $\int\int \cos(x)$ 。

```
f=sym('cos(x)');
int(f)         %求不定积分

ans =
sin(x)
int(f,0,pi/3)    %求定积分

ans =
1/2*3^(1/2)
int(f,'a','b')   %求定积分

ans =
sin(b)-sin(a)
int(int(f))      %求多重积分
```

```
ans =
-cos(x)
```

diff 和 int 命令，也可以直接对字符串 f 进行运算：

```
f='cos(x)';
```

【例 3.16 续】求符号矩阵$\begin{bmatrix} 2x & t^2 \\ t\sin(x) & e^x \end{bmatrix}$的积分。

```
syms t x
g=[2*x t^2;t*sin(x) exp(x)]  %创建符号矩阵

g =
[      2*x,      t^2]
[ t*sin(x),   exp(x)]
int(g)                   %对 x 求不定积分

ans =
[       x^2,     t^2*x]
[ -t*cos(x),    exp(x)]
int(g,'t')                  %对 t 求不定积分

ans =
[         2*x*t,        1/3*t^3]
[ 1/2*t^2*sin(x),       exp(x)*t]
int(g,sym('a'),sym('b'))    %对 x 求定积分

ans =
[           b^2-a^2,          t^2*(b-a)]
[ -t*cos(b)+t*cos(a),      exp(b)-exp(a)]
```

3.4.4 符号级数

1. symsum 函数

语法：

symsum(s,x,a,b)　　　　　　　　%计算表达式 s 的级数和

说明：x 为自变量，x 省略则默认为对自由变量求和；s 为符号表达式；[a,b]为参数 x 的取值范围。

【例 3.17】求级数$1+\frac{1}{2^2}+\frac{1}{3^2}+\ldots+\frac{1}{k^2}+\ldots$和 $1+x+x^2+\cdots+x^k+\cdots$的和。

```
syms x k
s1=symsum(1/k^2,1,10)       %计算级数的前 10 项和

s1 =
1968329/1270080
```

```
s2=symsum(1/k^2,1,inf)          %计算级数和

s2 =
1/6*pi^2
s3=symsum(x^k,'k',0,inf)        %计算对 k 为自变量的级数和

s3 =
-1/(x-1)
```

2. taylor 函数

语法：

taylor (F,x,n)　　　　　　%求泰勒级数展开

说明：x 为自变量，F 为符号表达式；对 F 进行泰勒级数展开至 n 项，参数 n 省略则默认展开前 5 项。

【例 3.17 续】求 e^x 的泰勒展开式为：$1+x+\frac{1}{2}\cdot x^2+\frac{1}{2\times 3}\cdot x^3+\ldots+\frac{1}{k!}\cdot x^{k-1}+\ldots$。

```
syms x
s1=taylor(exp(x),8)          %展开前 8 项

s1 =
1+x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5+1/720*x^6+1/5040*x^7
s2=taylor(exp(x))        %默认展开前 5 项

s2 =
1+x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5
```

3.5 符号积分变换

3.5.1 傅里叶(Fourier)变换及其反变换

fourier 变换和反变换可以利用积分函数 int 来实现，也可以直接使用 fourier 或 ifourier 函数实现。

1. fourier 变换

语法：

F=fourier(f,t ,w)　　　　　　%求时域函数 f(t)的 fourier 变换 F

说明：返回结果 F 是符号变量 w 的函数，当参数 w 省略，默认返回结果为 w 的函数；f 为 t 的函数，当参数 t 省略，默认自由变量为 x。

2. fourier 反变换

语法：

f=ifourier (F)　　　　　　%求频域函数 F 的 fourier 反变换 f(t)

f=ifourier (F,w,t)

说明：ifourier 函数的用法与 fourier 函数相同。

【例 3.18】计算 f(t)= $\frac{1}{t}$ 的 fourier 变换 F 以及 F 的 fourier 反变换。

```
syms t w
F=fourier(1/t,t,w)       %fourier 变换

F =
i*pi*(Heaviside(-w)-Heaviside(w))
f=ifourier(F,t)              %fourier 反变换

f =
1/t
f=ifourier(F)            %fourier 反变换默认 x 为自变量

f =
1/x
```

程序分析：其中 Heaviside(t)是单位阶跃函数 $\begin{cases} 1 & t \ge 0 \\ 0 & t < 0 \end{cases}$，函数名为数学家 Heaviside 的名字。

【例 3.18 续】单位阶跃函数的 fourier 变换。

```
fourier(sym('Heaviside(t)'))

ans =
pi*Dirac(w)-i/w
```

程序分析：Dirac 函数为单位脉冲函数。

3.5.2 拉普拉斯(Laplace)变换及其反变换

1. Laplace 变换

语法：

F=laplace(f,t,s)　　　%求时域函数 f 的 Laplace 变换 F

说明：返回结果 F 为 s 的函数，当参数 s 省略，返回结果 F 默认为's'的函数；f 为 t 的函数，当参数 t 省略，默认自由变量为't'。

【例 3.19】求 sin(at)和阶跃函数的 Laplace 变换。

```
syms a t s
F1=laplace(sin(a*t),t,s)       %求 sinat 的 Laplace 变换

F1 =
a/(s^2+a^2)
F2=laplace(sym('Heaviside(t)'))    %求阶跃函数的 Laplace 变换

F2 =
1/s
```

2. Laplace 反变换

语法：

f＝ilaplace(F,s,t)　　　　　　%求 F 的 Laplace 反变换 f

【例 3.19 续】求 $\frac{1}{s+a}$ 和 1 的 Laplace 反变换。

```
syms s a t
f1=ilaplace(1/(s+a),s,t)    %求 1/s+a 的 Laplace 反变换

f1 =
exp(-a*t)
f2=ilaplace(1,s,t)       %求 1 的 Laplace 反变换是脉冲函数

f2 =
Dirac(t)
```

3.5.3 Z 变换及其反变换

1. ztrans 函数

语法：

F＝ztrans(f,n, z)　　　　　　%求时域序列 f 的 Z 变换 F

说明：返回结果 F 是以符号变量 z 为自变量；当参数 n 省略，默认自变量为'n'；当参数 z 省略，返回结果默认为'z'的函数。

【例 3.20】求阶跃函数、脉冲函数和 e^{-at} 的 Z 变换。

```
syms a n z t
Fz1=ztrans(sym('Heaviside(t)'),n,z)   %求阶跃函数的 Z 变换

Fz1 =
Heaviside(t)*z/(z-1)
Fz2=ztrans(sym('Dirac(t)'),n,z)       %求脉冲函数的 Z 变换

Fz2 =
Dirac(t)*z/(z-1)
Fz3=ztrans(exp(-a*t),n,z)          %求 e-at 的 Z 变换

Fz3 =
exp(-a*t)*z/(z-1)
```

2. iztrans 函数

语法：

f＝iztrans(F,z,n)　　　　　%求 F 的 z 反变换 f

【例 3.20 续】用 Z 反变换验算阶跃函数、脉冲函数和 e^{-at} 的 Z 变换。

```
syms n z t
f1=iztrans(Fz1,z,n)

f1 =
```

```
Heaviside(t)
 f2=iztrans(Fz2,z,n)

f2 =
Dirac(t)
 f3=iztrans(Fz3,z,n)

f3 =
exp(-a*t)
```

3.6 符号方程的求解

3.6.1 代数方程

当方程不存在解析解又无其他自由参数时，MATLAB 可以用 solve 命令给出方程的数值解。

语法：

solve(‘eq’,’v’)　　　　　　%求方程关于指定变量的解

solve(‘eq1’, ’eq2’,’v1’,’v2’,…)　%求方程组关于指定变量的解

说明：eq 可以是含等号的符号表达式的方程，也可以是不含等号的符号表达式，但所指的仍是令 eq=0 的方程；当参数 v 省略时，默认为方程中的自由变量；其输出结果为结构数组类型。

【例 3.21】求方程 $ax^2+bx+c=0$ 和 sinx=0 的解。

```
 f1=sym('a*x^2+b*x+c')          %无等号

f1 =
a*x^2+b*x+c
 solve(f1)                    %求方程的解 x

ans =
[ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
 f2=sym('sin(x)')

f2 =
sin(x)
 solve(f2,'x')

ans =
0
```

程序分析：当 sinx=0 有多个解时，只能得出 0 附近的有限几个解。

【例 3.22】求三元非线性方程组 $\begin{cases} x^2+2x+1=0 \\ x+3z=4 \\ y*z=-1 \end{cases}$ 的解。

```
eq1=sym('x^2+2*x+1');
eq2=sym('x+3*z=4');
eq3=sym('y*z=-1');
[x,y,z]=solve(eq1,eq2,eq3)          %解方程组并赋值给 x,y,z
```

```
x =
-1
y =
-3/5
z =
5/3
```

程序分析：输出结果为“结构对象”，如果最后一句为“S=solve(eq1,eq2,eq3) ”，则结果为：

```
S =
    x: [1x1 sym]
    y: [1x1 sym]
    z: [1x1 sym]
```

3.6.2 符号常微分方程

MATLAB 提供了 dsolve 命令可以用于对符号常微分方程进行求解。

语法：

dsolve('eq','con','v') %求解微分方程

dsolve('eq1,eq2…','con1,con2…','v1,v2…') %求解微分方程组

说明：'eq'为微分方程；'con'是微分初始条件，可省略；'v'为指定自由变量，省略时则默认为 x 或 t 为自由变量；输出结果为结构数组类型。

- 当 y 是因变量时，微分方程'eq'的表述规定为：

y 的一阶导数 $\frac{dy}{dx}$ 或 $\frac{dy}{dt}$ 表示为 Dy；

y 的 n 阶导数 $\frac{d^n y}{dx^n}$ 或 $\frac{d^n y}{dt^n}$ 表示为 Dny。

- 微分初始条件'con'应写成'y(a)=b，Dy(c)=d'的格式；当初始条件少于微分方程数时，在所得解中将出现任意常数符 C1，C2……，解中任意常数符的数目等于所缺少的初始条件数。

【例 3.23】求微分方程 $x\frac{d^2y}{dx^2}-3\frac{dy}{dx}=x^2$，y(1)=0，y(0)=0 的解。

```
y=dsolve('x*D2y-3*Dy=x^2','x')                %求微分方程的通解
```

```
y =
-1/3*x^3+C1+C2*x^4
```

```
y=dsolve('x*D2y-3*Dy=x^2','y(1)=0,y(5)=0','x')   %求微分方程的特解
```

```
y =
-1/3*x^3+125/468+31/468*x^4
```

【例 3.24】求微分方程组 $\frac{dx}{dt}=y,\frac{dy}{dt}=-x$ 的解。

```
[x,y]=dsolve('Dx=y,Dy=-x')

x =
cos(t)*C1+sin(t)*C2
y =
-sin(t)*C1+cos(t)*C2
```

程序分析：默认的自由变量是 t，C1、C2 为任意常数，程序也可指定自由变量，结果相同：

```
[x,y]=dsolve('Dx=y,Dy=-x','t')
```

3.7 符号函数的可视化

3.7.1 符号函数的绘图命令

1. ezplot 和 ezplot3 命令

ezplot 命令是绘制符号表达式的自变量和对应各函数值的二维曲线，ezplot3 命令用于绘制三维曲线。

语法：

ezplot(F,[xmin,xmax],fig) %画符号表达式的图形

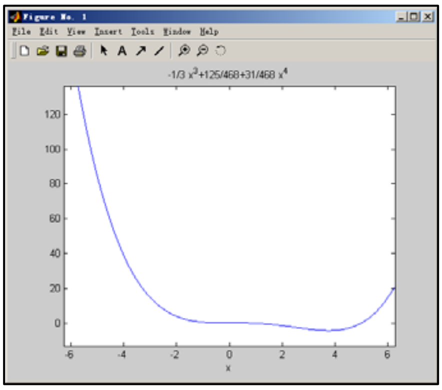
说明：F 是将要画的符号函数；[xmin,xmax]是绘图的自变量范围，省略时默认值为[－2л，2л]；fig 是指定的图形窗口，省略时默认为当前图形窗口。

【例 3.25】画出上一小节【例 3.23】中 y(x)特解的图形，如图 3.2 所示。

```
y =sym('-1/3*x^3+1/3*x^4')

y =
-1/3*x^3+1/3*x^4
ezplot(y)

ezplot(y,[0,100])           %绘制符号函数 y 在[0,100]中的图形
```



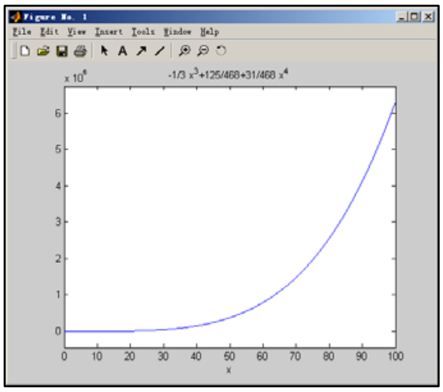


图 3.2 (a) y(x)在[-2π，2π]的图形　　　　(b) y(x)在[0，100]的图形

【例 3.26】用 ezplot3 绘制三维符号表达式曲线，如图 3.3 所示。

```
x=sym('sin(t)');
z=sym('t');
y=sym('cos(t)');
ezplot3(x,y,z,[0,10*pi],'animate')          %绘制 t 在[0,10*pi]范围的三维曲线
```

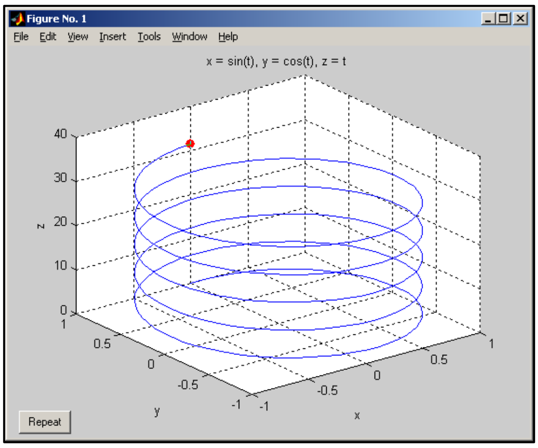


图 3.3 符号表达式绘制三维曲线

2. 其它绘图命令

MATLAB 提供的较常用绘图命令还有如表 3.3 所示。

表 3.3 符号表达式和字符串的绘图命令

命令名	含义	举例
ezcontour	画等高线	ezcontour('x*sin(t)',[-4,4])
ezcontourf	画带填充颜色等高线	ezcontourf('x*sin(t)',[-4,4])
ezmesh	画三维网线图	ezmesh('sin(x)*exp(-t)','cos(x)*exp(-t)','x',[0,2*pi])
ezmeshc	画带等高线的三维网线图	ezmeshc('sin(x)*t',[-pi,pi])
ezpolar	画极坐标图	ezpolar('sin(t)',[0,pi/2])
ezsurf	画三维曲面图	ezsurf('x*sin(t)','x*cos(t)','t',[0,10*pi])
ezsurfc	画带等高线的三维曲面图	ezsurfc('x*sin(t)','x*cos(t)','t',[0,pi,0,2*pi])

说明：这些命令的举例都是对字符串函数进行绘图，同样也可用于符号表达式绘图。

3.7.2 图形化的符号函数计算器

Symbolic Math Toolbox 还提供了另一种符号计算方式即图形化的符号函数计算器，由 funtool.m 文件生成。在 MATLAB 命令窗口输入命令“funtool”，就会出现该图形化函数计算器。如图 3.4 所示。

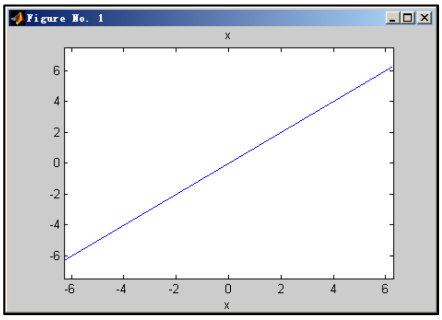
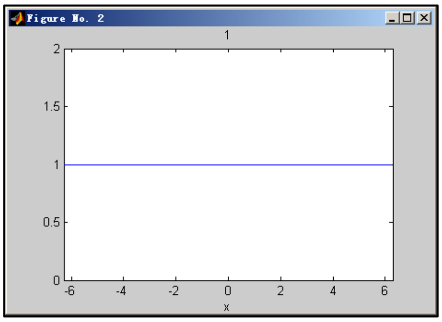


图 3.4 (a) Figure No. 1 窗口



(b) Figure No. 2 窗口

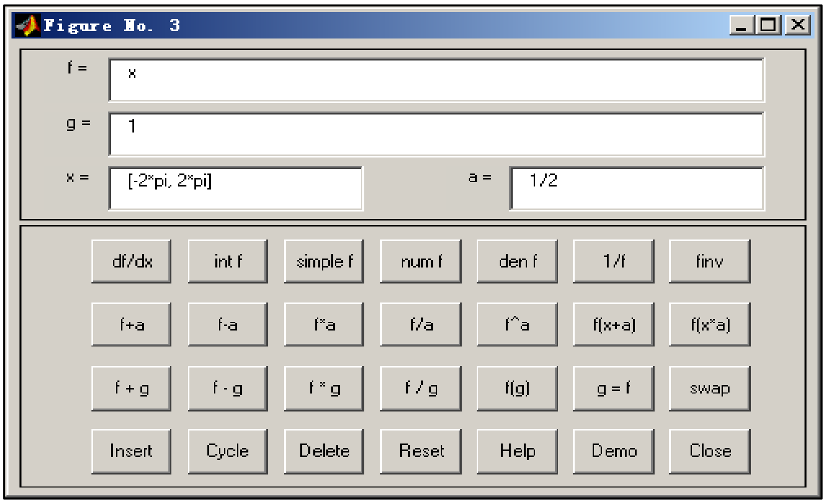


图 3.4 (c) Figure No. 3 窗口

在图形化函数计算器中可以方便地查看函数的计算结果和显示的曲线。

3.8 Maple 函数的使用

MATLAB 的符号运算实际上是由 Maple 软件所支持的符号数学工具箱完成的，Maple 软件有 2000 多条符号运算命令。

3.8.1 访问 Maple 函数

1. maple 函数

maple 函数用于进行符号运算，并将计算结果返回到 MATLAB 的工作空间。

语法：

maple(MapleStatement)　　　　%运行 Maple 格式的语句 MapleStatement

maple(fun,arg1,arg2…)　　　　%运行以 arg1,arg2…为参数的 Maple 的 fun 函数

【例 3.27】利用 Maple 的 discrim 函数，计算多项式的判别式。

```
maple('discrim(a*x^2+b*x+c,x)')

ans =
-4*a*c+b^2
```

也可以用以下语句实现：

```
syms a b c x
maple('discrim',a*x^2+b*x+c,'x')

ans =
-4*a*c+b^2
```

程序分析：MapleStatement 必须符合 Maple 语法规则，因此一般应先用 mhelp 命令获得 maple 函数的命令格式。

2. mfun 函数

mfun 函数用于对 Maple 中的经典函数进行数值运算。

语法：

mfun('fun',p1,p2,…)

说明：'fun'为函数名；p1,p2,…为函数的参数。

【例 3.28】利用 gcd 函数计算最大公约数。

```
mfun('gcd',20,30)

ans =
    10
```

3.8.2 获得 Maple 的帮助

1. mfunlist 命令

mfunlist 命令用来列出能被“mfun”命令计算的经典特殊 Maple 函数。

2. mhelp 命令

mhelp 命令用来寻求关于 Maple 库函数及其调用方法的帮助：

(1) 使用“mhelp index”可以查看 Maple 的索引目录。

(2) 使用“mhelp index [分类名]”可以进一步深入查看 Maple 的某个具体类别。当输入“mhelp index”命令会出现如表 3.4 所示的类别名。

表 3.4 Maple 的函数类别

类别名	说明
expression	表达式描述命令集
function	函数命令集

【例 3.28 续】查看求最大公约数的函数 gcd。

```
mhelp gcd
```

第 4 章 MATLAB 计算的可视化和 GUI 设计

MATLAB 具有非常强大的二维和三维绘图功能，尤其擅长于各种科学运算结果的可视化。

4.1 二维曲线的绘制

4.1.1 基本绘图命令 plot

1. plot(x) 绘制 x 向量曲线

plot 命令是 MATLAB 中最简单而且使用最广泛的一个绘图命令，用来绘制二维曲线。

语法：

plot(x)　　　%绘制以 x 为纵坐标的二维曲线

plot(x,y)　　%绘制以 x 为横坐标 y 为纵坐标的二维曲线

说明：x 和 y 可以是向量或矩阵。

【例 4.1】用 plot(x)命令画直线，如图 4.1 所示。

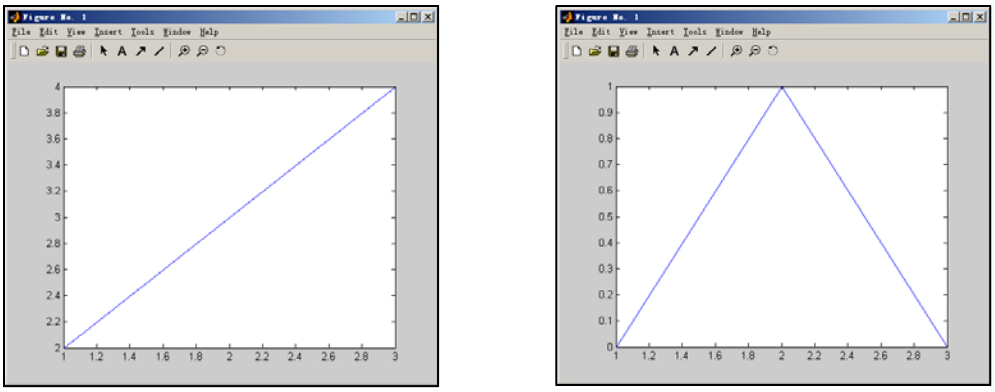


图 4.1 (a) x1 曲线　　　　(b) x2 曲线

```
x1=[1 2 3]

x1 =
     1     2     3

plot(x1)
```

```
x2=[0 1 0]

x2 =
     0     1     0
plot(x2)
```

2. plot(x,y) 绘制向量 x 和 y 的曲线

【例 4.2】绘制正弦曲线 y=sin(x)和方波曲线，如图 4.2 所示。

```
x1=0:0.1:2*pi;
y1=sin(x1);        %y1 为 x1 的正弦函数
plot(x1,y1);
x2=[0 1 1 2 2 3 ];
y2=[1 1 0 0 1 1 ];
plot(x2,y2);
axis([0 4 0 2])           %将坐标轴范围设定为 0-4 和 0-2
```

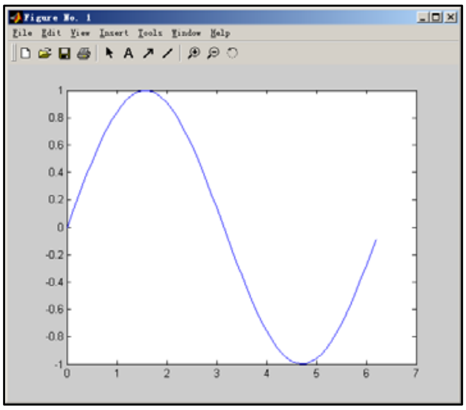
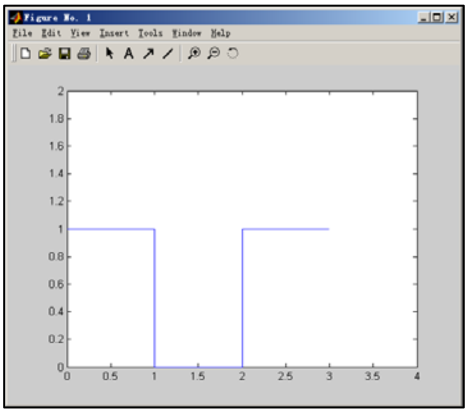


图 4.2 (a) 正弦曲线



(b) 方波曲线

3. plot(x) 绘制矩阵 x 的曲线

【例 4.3】矩阵图形的绘制，如图 4.3 所示。

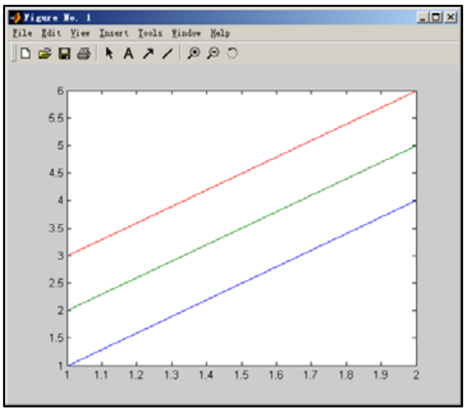
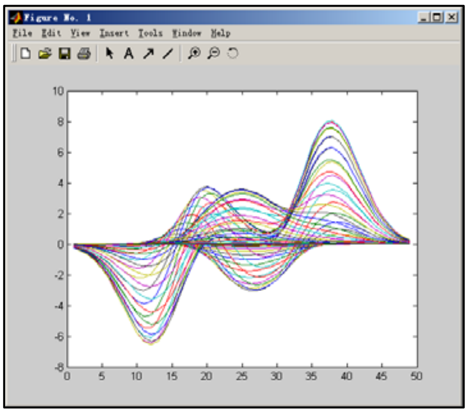


图 4.3 (a) x1 曲线



(b) x2 曲线

```
x1=[1 2 3;4 5 6];
plot(x1);
x2=peaks;              %产生一个 49*49 的矩阵
plot(x2);
```

程序分析：a 图中有三条曲线而不是两条曲线，因为矩阵 x1 有三列，每列向量画一条曲线；b 图为由 peaks 函数生成的一个 49×49 的二维矩阵，因此产生 49 条曲线。

4. plot(x,y)绘制混合式曲线

当 plot(x,y)命令中的参数 x 和 y 是向量或矩阵时，分别有以下几种情况：

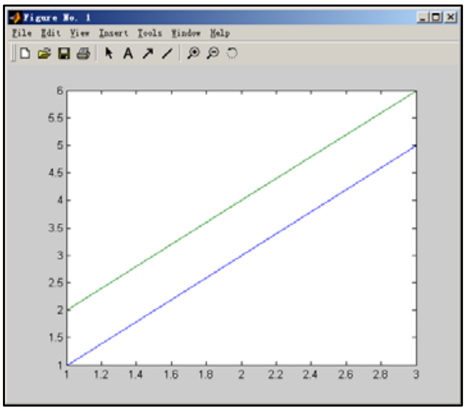
- 如果 x 是向量，而 y 是矩阵，则 x 的长度与矩阵 y 的行数或列数必须相等，如果 x 的长度与 y 的行数相等，则向量 x 与矩阵 y 的每列向量对应画一条曲线；如果 x 的长度与 y 的列数相等，向量 x 与 y 的每行向量画一条曲线，如果 y 是方阵，则 x 和 y 的行数和列数都相等，将向量 x 与矩阵 y 的每列向量画一条曲线；
- 如果 x 是矩阵，而 y 是向量，则 y 的长度必须等于 x 的行数或列数，绘制的方法与前一种相似；
- 如果 x 和 y 都是矩阵，则大小必须相同，矩阵 x 的每列和 y 的每列画一条曲线。

【**例 4.4**】混合式图形的绘制，如图 4.4 所示。

```
x1=[1 2 3];
y1=[1 2 3;4 5 6]

y1 =
     1     2     3
     4     5     6
plot(x1,y1)      %每行一条曲线
y2=[1 2 ;3 4; 5 6]

y2 =
     1     2
     3     4
     5     6
plot(x1,y2)      %每列一条曲线
plot(y1,x1)
plot(y2,x1)
x2=[1 1 1;2 2 2]
```



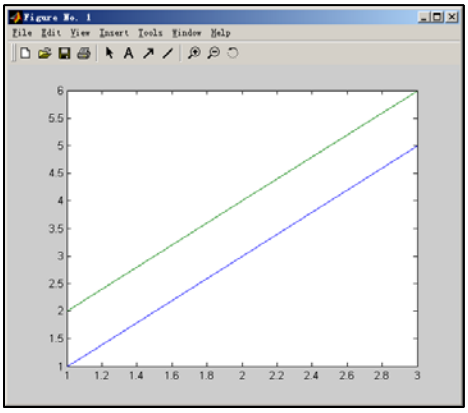
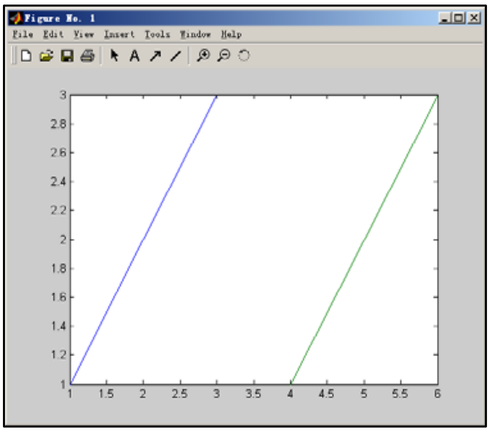


图 4.4 (a) (x1,y1)曲线　　　　(b) (x2,y1)曲线



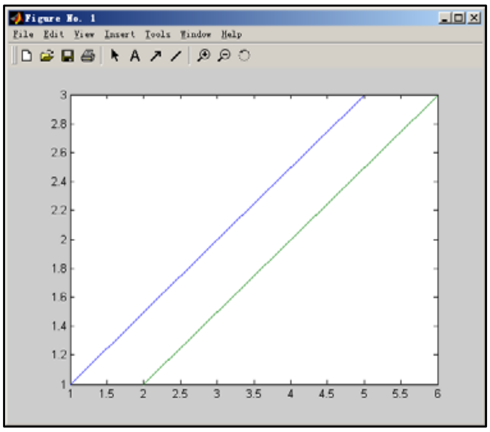


图 4.4 (c) (y1,x1)曲线　　　　(d) (y2,x1)曲线

```
x2 =
     1     1     1
     2     2     2
 plot(x2,y1)        %按列与列对应的方式
```

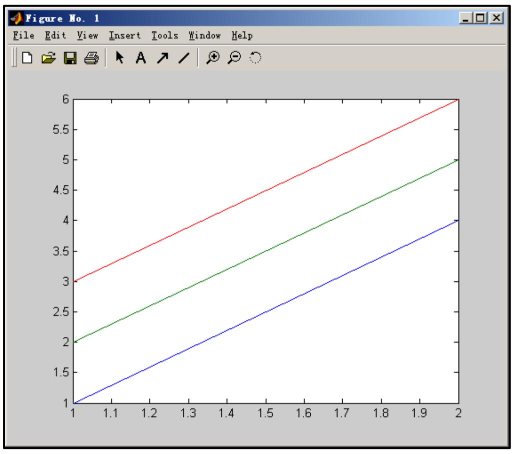


图 4.4 (e) (x2,y1)曲线

5. plot(z)绘制复向量曲线

【例 4.4 续】下面的程序画出的曲线和图 4.4(e)中的相同。

```
z1=x2+i*y1

z1 =
   1.0000 + 1.0000i   1.0000 + 2.0000i   1.0000 + 3.0000i
   2.0000 + 4.0000i   2.0000 + 5.0000i   2.0000 + 6.0000i
plot(z1)            %以实部为横坐标，虚部为纵坐标
```

6. plot(x1,y1,x2,y2,…)绘制多条曲线

plot 命令还可以同时绘制多条曲线，用多个矩阵对为参数，MATLAB 自动以不同的颜色绘制不同曲线。每一对矩阵(xi,yi)均按照前面的方式解释，不同的矩阵对之间，其维数可以不同。

【例 4.5】绘制三条曲线，如图 4.5 所示。

```
x=0:0.1:2*pi;
plot(x,sin(x),x,cos(x),x,sin(3*x))  %画三条曲线
```

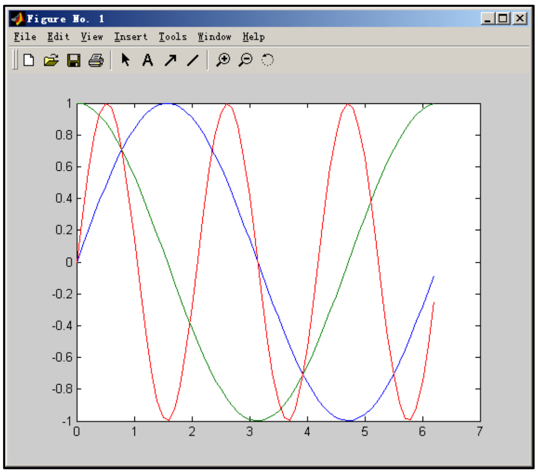


图 4.5　三条曲线

4.1.2 绘制曲线的一般步骤

表 4.1 为绘制二维、三维图形一般步骤的归纳。

表 4.1　绘制二维、三维图形的一般步骤

步骤	内容
1	曲线数据准备： 对于二维曲线，横坐标和纵坐标数据变量； 对于三维曲面，矩阵参变量和对应的函数值。
2	指定图形窗口和子图位置： 默认时，打开 Figure No.1 窗口或当前窗口、当前子图； 也可以打开指定的图形窗口和子图。
3	设置曲线的绘制方式： 线型、色彩、数据点形。
4	设置坐标轴： 坐标的范围、刻度和坐标分格线

5	图形注释： 图名、坐标名、图例、文字说明
6	着色、明暗、灯光、材质处理(仅对三维图形使用)
7	视点、三度(横、纵、高)比(仅对三维图形使用)
8	图形的精细修饰(图形句柄操作)： 利用对象属性值设置； 利用图形窗工具条进行设置。

说明：

- 步骤 1 和 3 是最基本的绘图步骤，如果利用 MATLAB 的默认设置通常只需要这两个基本步骤就可以基本绘制出图形，而其他步骤并不完全必需。
- 步骤 2 一般在图形较多的情况下，需要指定图形窗口、子图时使用。
- 除了步骤 1、2、3 的其他步骤用户可以根据自己需要改变前后次序。

4.1.3 多个图形绘制的方法

1. 指定图形窗口

如果需要多个图形窗口同时打开时，可以使用 figure 语句。

语法：

figure(n)　　　　%产生新图形窗口

说明：如果该窗口不存在，则产生新图形窗口并设置为当前图形窗口，该窗口名为“Figure No.n”，而不关闭其它窗口。

2. 同一窗口多个子图

如果需要在同一个图形窗口中布置几幅独立的子图，可以在 plot 命令前加上 subplot 命令来将一个图形窗口划分为多个区域，每个区域一幅子图。

语法：

subplot(m,n,k)　　　　%使(m×n)幅子图中的第 k 幅成为当前图

说明：将图形窗口划分为 m×n 幅子图，k 是当前子图的编号，“,”可以省略。子图的序号编排原则是：左上方为第 1 幅，先向右后向下依次排列，子图彼此之间独立。

【例 4.6】用 subplot 命令画四个子图，如图 4.6 所示。

```
x=0:0.1:2*pi;
subplot(2,2,1)          %分割为 2*2 个子图，左上方为当前图
plot(x,sin(x))
subplot(2,2,2)          %右上方为当前图
plot(x,cos(x))
subplot(2,2,3)          %左下方为当前图
plot(x,sin(3*x))
subplot(224)            %右下方为当前图，省略逗号
plot(x,cos(3*x))
```

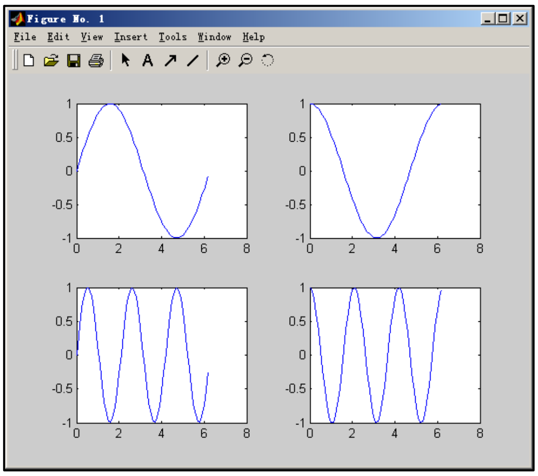


图 4.6　四个子图

如果在使用绘图命令之后，想清除图形窗口画其它图形，应使用“clf”命令清图形窗。

```
clf                    %清除子图
```

3. 同一窗口多次叠绘

为了在一个坐标系中增加新的图形对象，可以用“hold”命令来保留原图形对象。

语法：

hold on　　　　%使当前坐标系和图形保留

hold off　　　　%使当前坐标系和图形不保留

hold　　　　　%在以上两个命令中切换

说明：在设置了“hold on”后，如果画多个图形对象，则在生成新的图形时保留当前坐标系中已存在的图形对象，MATLAB 会根据新图形的大小，重新改变坐标系的比例。

【例 4.7】在同一窗口画出函数 sinx 在区间[0 2π]的曲线和 cosx 在区间[-π π]的曲线，如图 4.7(a)所示。

```
x1=0:0.1:2*pi;
plot(x1,sin(x1))
hold on
x2=-pi:.1:pi;
plot(x2,cos(x2))
```

程序分析：坐标系的范围由 0～2π转变为-π～2π。

4. 双纵坐标图

语法：

plotyy(x1,y1,x2,y2)　　%以左、右不同纵轴绘制两条曲线

说明：左纵轴用于(x1,y1)数据，右纵轴用于(x2,y2)数据来绘制两条曲线。坐标轴的范围、刻度都自动产生。

【例 4.7 续】用 plotyy 函数实现在同一图形窗口绘制两条曲线，如图 4.7(b)所示。

```
plotyy(x1,sin(x1),x2,cos(x2))
```

程序分析：plotyy 函数用不同颜色绘制两条曲线，左右两边使用两个纵坐标轴，横坐标从-π～2π。

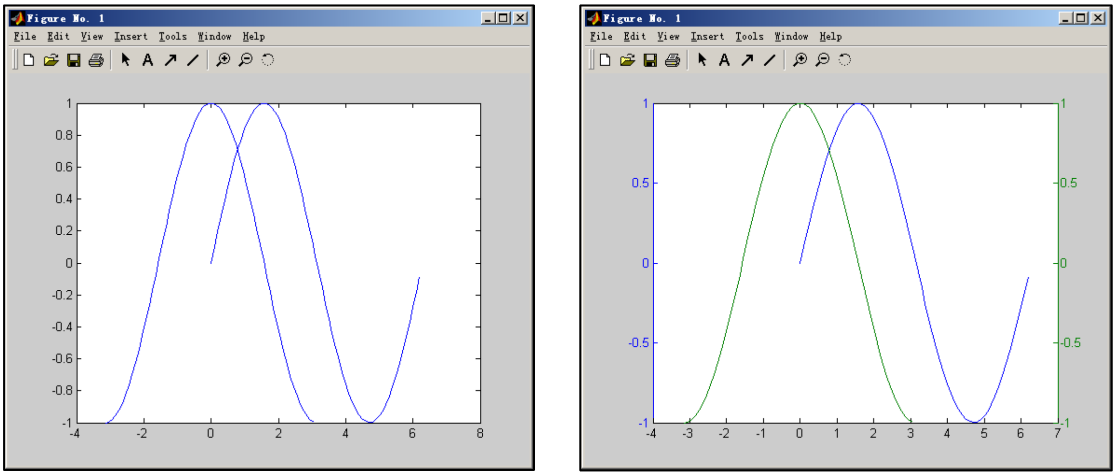


图 4.7 (a) 用 **hold on** 在同一窗口画出两条曲线　　　(b) 用 **plotyy** 在同一窗口画出两条曲

4.1.4 曲线的线型、颜色和数据点形

plot 命令还可以设置曲线的线段类型、颜色和数据点形等，如表 4.2 所示。

表 4.2　线段、颜色与数据点形

颜色		数据点间连线		数据点形	
类型	符号	类型	符号	类型	符号
黄色	y(Yellow)	实线(默认)	-	实点标记	.
品红色(紫色)	m(Magenta)	点线	:	圆圈标记	o
青色	c(Cyan)	点划线	-.	叉号形×	x
红色	r(Red)	虚线	--	十字形＋	+
绿色	g(Green)			星号标记＊	*
蓝色	b(Blue)			方块标记□	s
白色	w(White)			钻石形标记◇	d
黑色	k(Black)			向下的三角形标记	v
				向上的三角形标记	^
				向左的三角形标记	<
				向右的三角形标记	>
				五角星标记☆	p
				六连形标记	h

语法：

plot(x,y,s)

说明：x 为横坐标矩阵，y 为纵坐标矩阵，s 为类型说明字符串参数；s 字符串可以是线段类型、颜色和数据点形三种类型的符号之一，也可以是三种类型符号的组合。

【例 4.8】用不同线段类型、颜色和数据点形画出 sinx 和 cosx 曲线，如图 4.8 所示。

```
x=0:0.1:2*pi;
```

```
plot(x,sin(x),'r-.')          %用红色点划线画出曲线
hold on
plot(x,cos(x),'b:o')          %用蓝色圆圈画出曲线，用点线连接
```

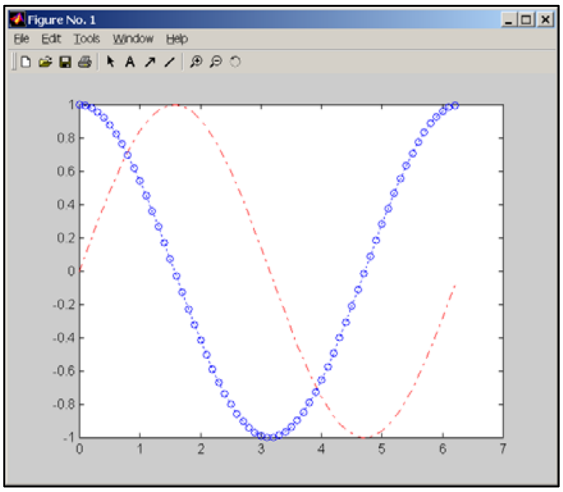


图 4.8 在同一窗口画出两条曲线

4.1.5 设置坐标轴和文字标注

1. 坐标轴的控制

用坐标控制命令 axis 来控制坐标轴的特性，表 4.3 列出其常用控制命令。

表 4.3　常用的坐标控制命令

命令	含义	命令	含义
axis auto	使用默认设置	axis equal	纵、横轴采用等长刻度
axis manual	使当前坐标范围不变	axis fill	在 manual 方式下起作用，使坐标充满整个绘图区
axis off	取消轴背景	axis image	纵、横轴采用等长刻度，且坐标框紧贴数据范围
axis on	使用轴背景	axis normal	默认矩形坐标系
axis ij	矩阵式坐标，原点在左上方	axis square	产生正方形坐标系
axis xy	普通直角坐标，原点在左下方	axis tight	把数据范围直接设为坐标范围
axis([xmin,xmax, ymin,ymax])	设定坐标范围，必须满足 xmin<xmax,ymin<ymax，可以取 inf 或-inf。	axis vis3d	保持高宽比不变，用于三维旋转时避免图形大小变化

2. 分格线和坐标框

(1) 使用 grid 命令显示分格线

语法：

grid on　　　　%显示分格线
grid off　　　　%不显示分格线
grid　　　　　%在以上两个命令间切换

说明：不显示分格线是 MATLAB 的默认设置。分格线的疏密取决于坐标刻度，如果要改变分格线的疏密，必须先定义坐标刻度。

(2) 使用 box 命令显示坐标框

语法：

box on %使当前坐标框呈封闭形式

box off %使当前坐标框呈开启形式

box %在以上两个命令间切换

说明：在默认情况下，所画的坐标框呈封闭形式。

【例 4.9】在两个子图中使用坐标轴、分格线和坐标框控制，如图 4.9 所示。

```
x=0:0.1:2*pi;
subplot(2,1,1)
plot(sin(x),cos(x))
axis equal          %纵、横轴采用等长刻度
grid on             %加分格线
subplot(2,1,2)
plot(x,exp(-x))
axis([0,3,0,2])       %改变坐标轴范围
```

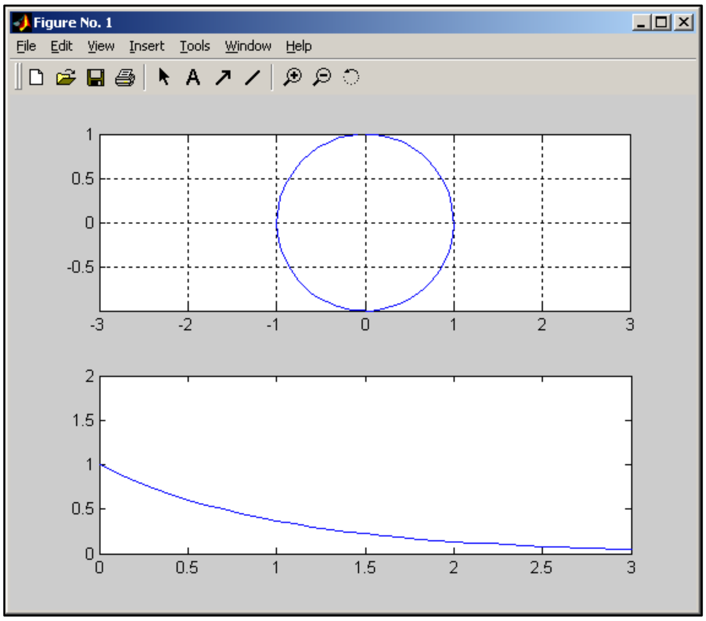


图 4.9 用坐标轴、分格线和坐标框控制

3. 文字标注

(1) 添加图名

语法：

title(s) %书写图名

说明：s 为图名，为字符串，可以是英文或中文。

(2) 添加坐标轴名

语法：

xlabel(s) %横坐标轴名

ylabel(s) %纵坐标轴名

(3) 添加图例

语法：

legend(s,pos) %在指定位置建立图例

legend off %擦除当前图中的图例

说明：参数 s 是图例中的文字注释，如果多个注释则可以用’s1’,’s2’,…的方式；参数 pos 是图例在图上位置的指定符，它的取值如表 4.4 所示。

表 4.4 pos 取值所对应的图例位置

pos 取值	0	1	2	3	4	−1
图例位置	自动取最佳位置	右上角(默认)	左上角	左下角	右下角	图右侧

用 legend 命令在图形窗口中产生图例后，还可以用鼠标对其进行拖拉操作，将图例拖到满意的位置。

(4) 添加文字注释

语法：

text(xt,yt,s) %在图形的(xt,yt)坐标处书写文字注释

【例 4.10】在图形窗口中添加文字注释，如图 4.10 所示。

```
x=0:0.1:2*pi;
plot(x,sin(x))
hold on
plot(x,cos(x),'ro')
title('y1=sin(x),y2=cos(x)')    %添加标题
xlabel('x')                        %添加横坐标名
legend('sin(x)','cos(x)',4)     %在右下角添加图例
text(pi,sin(pi),'x=\pi')        %在 pi,sin(pi)处添加文字注释
```

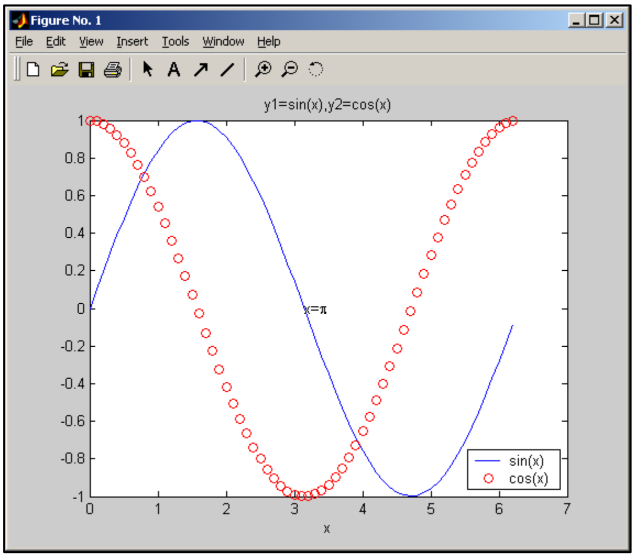


图 4.10 添加图形标注

4. 特殊符号

表 4.5 图形标识用的希腊字母、数学符号和特殊字符

类别	命令	字符	命令	字符	命令	字符	命令	字符
希	\ alpha	α	\ eta	η	\ nu	ν	\ upsilon	υ
腊	\ beta	β	\ theta	θ	\ xi	ξ	\ Upsilon	ϒ

字母	\ epsilon	ε	\ Theta	Θ	\ Xi	Ε	\ phi	ϕ
	\ gamma	γ	\ iota	ι	\ pi	π	\ Phi	Φ
	\ Gamma	Γ	\ zeta	ζ	\ Pi	Π	\ chi	χ
	\ delta	δ	\ kappa	κ	\ rho	ρ	\ psi	ψ
	\ Delta	Δ	\ mu	μ	\ tau	τ	\ Psi	Ψ
	\ omega	ω	\ lambda	λ	\ sigma	σ		
	\ Omega	Ω	\ Lambda	Λ	\ Sigma	Σ		
数学符号	\approx	≈	\oplus	≡	\neq	≠	\leq	≤
	\geq	≥	\pm	±	\times	×	\div	÷
	\int	∫	\exists	∝	\infty	∞	\in	∈
	\sim	≌	\forall	∼	\angle	∠	\perp	⊥
	\cup	∪	\cap	∩	\vee	∨	\wedge	∧
	\surd	√	\otimes	⊗	\oplus	⊕		
箭头	\uparrow	↑	\downarrow	↓	\rightarrow	→	\leftarrow	←
	\leftrightarrow	↔	\updownarrow	↕				

如果需要对文字进行上下标设置，或设置字体大小，则必须在文字标识前先使用表 4.6 中所示的设置值。

表 4.6　文字设置

命令	含义
\fontname{s}	字体的名称，s 为 Times New Roman 、Courier、宋体等。
\fontsize{n}	字号大小，n 为正整数，默认为 10(points)。
\s	字体风格，s 可以为 bf(黑体)、it(斜体一)、sl(斜体二)、rm(正体)等。
^{s}	将 s 变为上标
_{s}	将 s 变为下标

【例 4.11】在 MATLAB 的图形窗口中写出标题为表达式 $y(\omega)=\int_0^{\infty} y(t)e^{-j\omega t}dt$，字体大小为 16 号，如图 4.11 所示。

```
    figure(1)
title('\fontsize{16}y(\omega)=\int^{\infty}_{0}y(t)e^{-j\omegat}dt')
```



图 4.11 特殊字符

4.1.6 交互式图形命令

1. ginput 命令

ginput 命令是从图上获取数据。

语法：

[x,y]=ginput(n)　　　%用鼠标从图形上获取 n 个点的坐标(x,y)

说明：参数 n 应为正整数，是通过鼠标从图上获得数据点的个数；x、y 用来存放所取点的坐标。

2. gtext 命令

gtext 命令是把字符串放置到图形中鼠标所指定的位置上。

语法：

gtext('s')　　　%用鼠标把字符串放置到图形上

说明：如果参数 s 是单个字符串或单行字符串矩阵，那么一次鼠标操作就可把全部字符以单行形式放置在图上；如果参数 s 是多行字符串矩阵，那么每操作一次鼠标，只能放置一行字符串，需要通过多次鼠标操作，把一行一行字符串放在图形的不同位置。

【例 4.12】在 y=sin(x)的图形中将(π，0)和(2π，0)点的坐标取出，并在(2π，0)点写“2π”字符串。

```
 x=0:0.1:2*pi;
 plot(x,sin(x))
 [m,n]=ginput(2)      %取两点坐标
m =
    3.1532
    6.2984
n =
   -0.0029
   -0.0088
 gtext('2\pi')         %写 2π
```

程序分析：由于鼠标所取点的位置有些偏差，因此 ginput 命令获取的坐标并不是精确在(π，0)和(2π，0)点上；gtext 命令在图中鼠标单击处写了“2π”字符串。

4.2 MATLAB 的三维图形绘制

4.2.1 绘制三维线图命令 plot3

plot3 是用来绘制三维曲线的，它的使用格式与二维绘图的 plot 命令很相似。

语法：

plot3(x,y,z, 's')　　　%绘制三维曲线

plot3(x1,y1,z1, 's1',x2,y2,z2, 's2',…)　%绘制多条三维曲线

说明：当 x、y、z 是同维向量时，则绘制以 x、y、z 元素为坐标的三维曲线；当 x、y、z 是同维矩阵时，则绘制三维曲线的条数等于矩阵的列数。s 是指定线型、色彩、数据点形的字符串。

【例 4.13】三维曲线绘图，如图 4.12 所示。

```
x=0:0.1:20*pi;
plot3(x,sin(x),cos(x))                    %按系统默认设置绘图
```

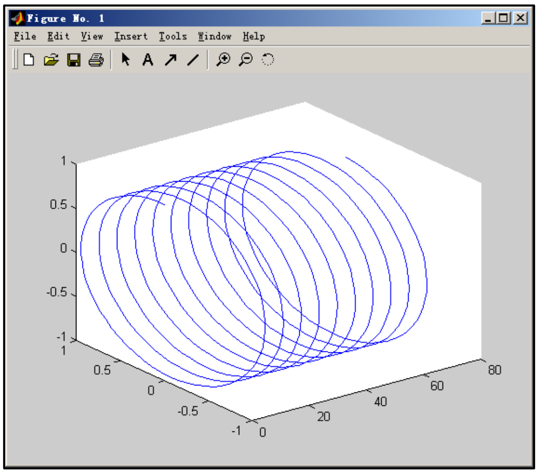


图 4.12 三维曲线

4.2.2 绘制三维网线图和曲面图

1. meshgrid 命令

为了绘制三维立体图形，MATLAB 的方法是将 x 方向划分为 m 份，将 y 方向划分为 n 份，meshgrid 命令是以 x、y 向量为基准，来产生在 x-y 平面的各栅格点坐标值的矩阵。

语法：

[X,Y]＝meshgrid(x,y)

说明：X、Y 是栅格点的坐标，为矩阵；x、y 为向量。

例如，将 x(1×m)向量和 y(1×n)向量转换为(n×m)的矩阵：

```
x=[1 2 3 4];
y=[5 6 7];
[xx,yy]=meshgrid(x,y)

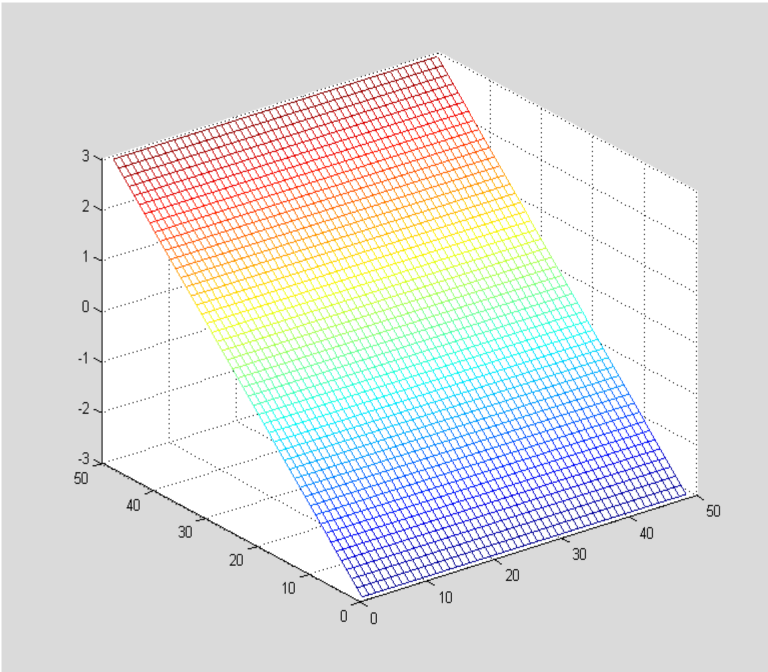
xx =
     1     2     3     4
     1     2     3     4
     1     2     3     4
yy =
     5     5     5     5
     6     6     6     6
     7     7     7     7
```

【例 4.14】使用 peaks 函数来测试 meshgrid 命令，并使用 mesh 命令来查看 meshgrid 的输出。

MATLAB 提供了 peaks 函数，在下面的图 4.13 中可以看到。其 x 和 y 坐标分别为在[-3 3]范围内的 49×49 的矩阵，z 坐标与 x、y 的关系为：

z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
  - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
  - 1/3*exp(-(x+1).^2 - y.^2)

```
x=linspace(-3,3,49);
y=linspace(-3,3,49);
[xx,yy]=meshgrid(x,y) ;          %产生 49*49 的栅格点坐标
mesh(xx)                        %查看 xx 的网线图
mesh(yy)
```



xx 和 yy 分别为 49×49 的矩阵，如图 4.13 为 xx 和 yy 的网状图。

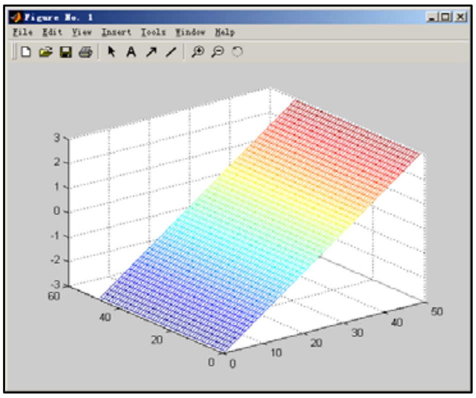
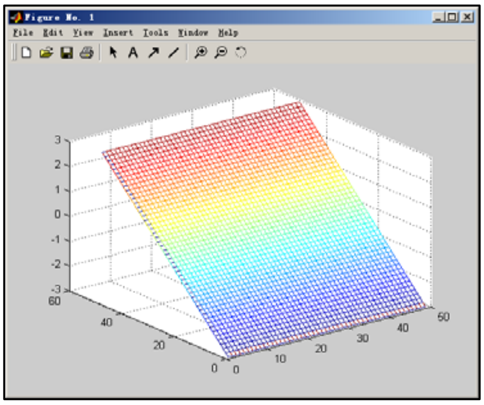


图 4.13 (a) xx 的网格图



(b) yy 的网格图

```
zz=3*(1-xx).^2.*exp(-(xx.^2) - (yy+1).^2) ...
  - 10*(xx/5 - xx.^3 - yy.^5).*exp(-xx.^2-yy.^2) ...
  - 1/3*exp(-(xx+1).^2 - yy.^2);       %产生 peaks 函数
plot3(xx,yy,zz)
```

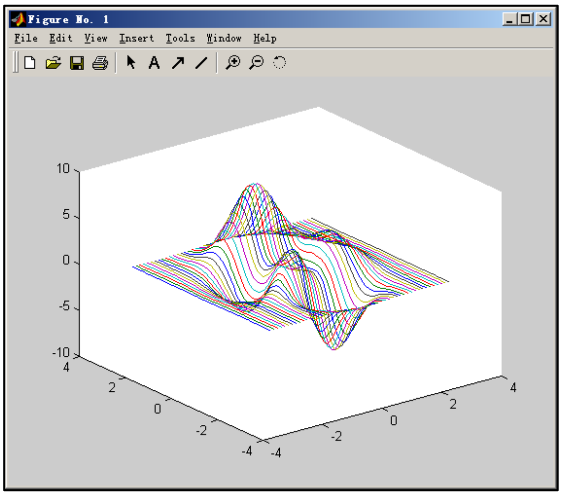


图 4.14　**peaks** 函数的三维线图

2. 三维网线图

语法：

mesh(z)　　　　　　　　　%画三维网线图

mesh(x,y,z,c)

说明：当只有参数 z 时，以 z 矩阵的行下标作为 x 坐标轴，把 z 的列下标当作 y 坐标轴；x、y 分别为 x、y 坐标轴的自变量；当有 x、y、z 参数时，c 是指定各点的用色矩阵，当 c 省略时默认用色矩阵是 z 的数据。如果 x、y、z、c 四个参数都有，则应该都是维数相同的矩阵。

【例 **4.14** 续】用 mesh 查看 peaks 函数的三维网线图，如图 4.15 所示。

```
mesh(xx,yy,zz)
```

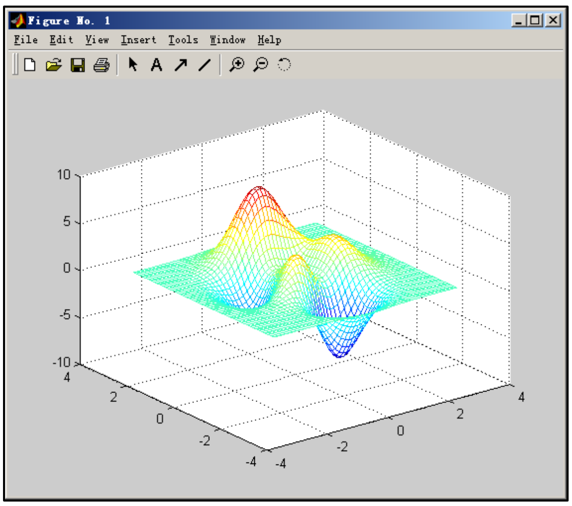


图 4.15　**peaks** 函数的三维网线图

3. 三维曲面图

语法：

surf (z)　　　　　　　%画三维曲面图

surf (x,y, z,c)

说明：参数设置与 mesh 命令相同，c 也可以省略。

【例 4.14 续】用 surf 查看 peaks 函数的三维曲面图，如图 4.16 所示。

```
surf (xx,yy,zz)
```

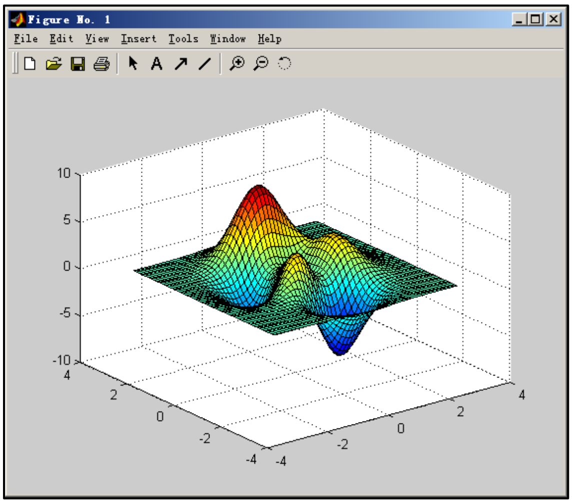


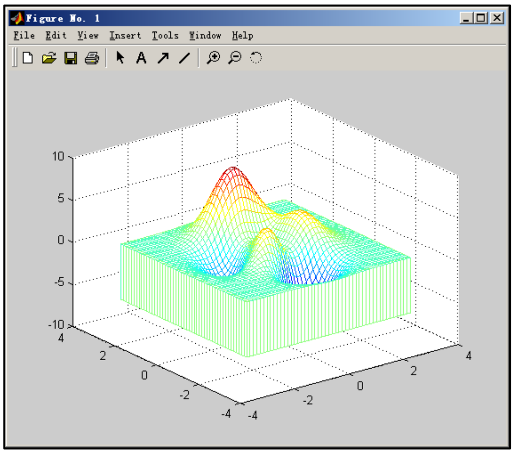
图 4.16 peaks 函数的三维曲面图

4. 其它立体网线图和曲面图

meshc 命令为立体网状图加等高线；meshz 为立体网状图加“围裙”。

【例 4.14 续】用 meshz 和 meshc 查看 peaks 函数的三维曲面图，如图 4.17 所示。

```
meshz(xx,yy,zz)
meshc(xx,yy,zz)
```



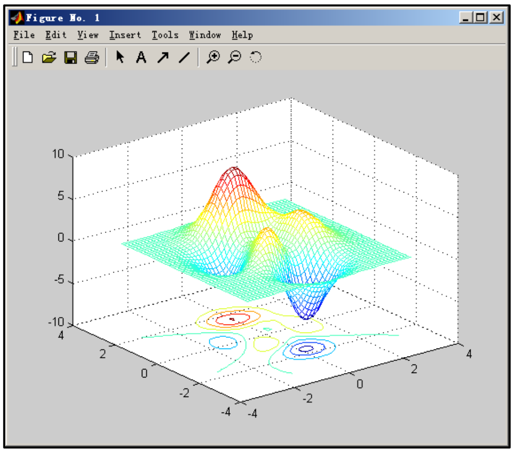


图 4.17 (a) peaks 函数的曲面加“围裙”图 (b) peaks 函数的曲面图加等高线

4.2.3 立体图形与图轴的控制

1．网格的隐藏

如果要使被遮盖的网格也能呈现出来，可用“hidden off”命令。

语法：

hidden off %显示被遮盖的网格

hidden on %隐藏被遮盖的网格

【例 4.15】显示被遮盖的网格，如图 4.18 所示。

```
[x,y,z]=peaks;          %peaks 函数
mesh(x,y,z)           %绘制曲面图
hidden off              %显示网格
```

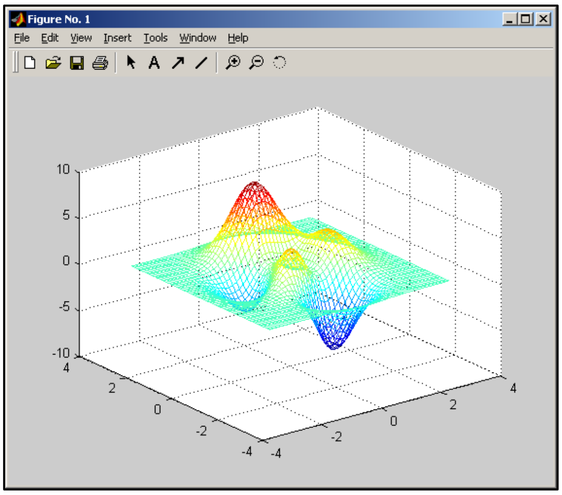


图 4.18 显示网线的 peaks 函数

2．改变视角

三维图形的观测角度不同则显示也不同，如果要改变观测角度，可用“view”命令。

语法：

view([az,el]) %通过方位角和俯仰角改变视角

view([vx,vy,vz]) %通过直角坐标改变视角

说明：az 表示方位角，el 表示俯仰角；vx、vy、vz 表示直角坐标。

【例 4.15 续】改变 peaks 函数的视角，如图 4.19 所示。

```
view(0,0)
view(0,90)
view(-37.5,30)        %恢复原视角
```

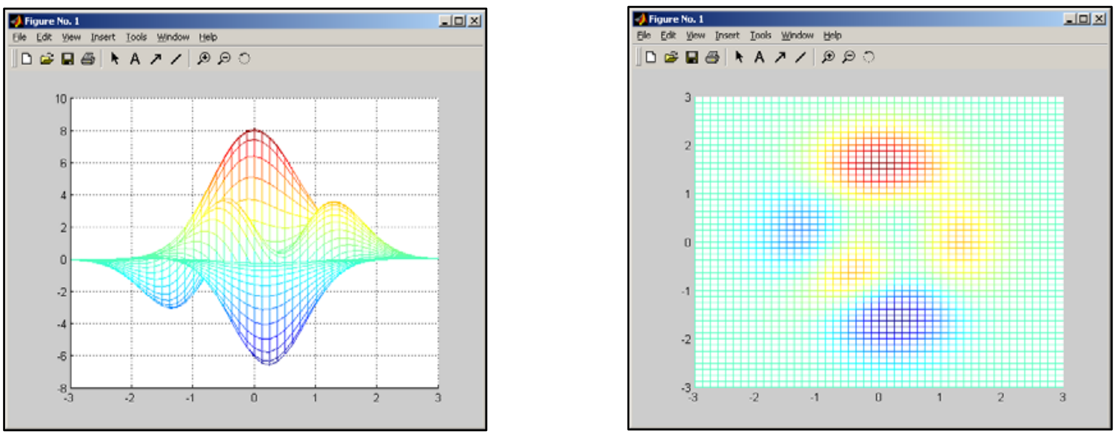


图 4.19 (a) 视角为(0,0)的 peaks 函数 **(b) 视角为(0,90)的 peaks 函数**

程序分析：视角为(0,0)，得到一个(x,z)的二维图形效果；视角为(0,90)，得到一个(x,y)的二维图形效果。

3．曲面的镂空

【例 4.15 续】对 peaks 函数曲面实现镂空效果，如图 4.20 所示。

```
z(10:20,10:20)=nan;      %将一部分数值用 nan 替换
surf(x,y,z)              %画曲面图
```

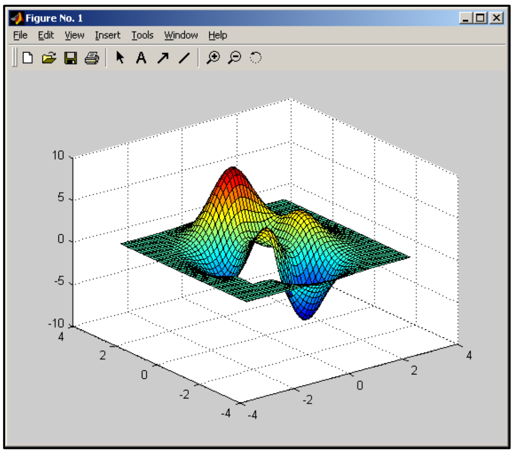


图 4.20 peaks 函数

4.2.4 色彩的控制

1．色图(colormap)

(1) RGB 三元组

RGB 三元行数组表示一种色彩，数组元素 R、G、B 在 0～1 之间，分别表示红、绿、蓝基色的相对亮度，如表 4.7 所示。

表 4.7　常用颜色的 RGB 成分

颜色	RGB 成分		
	Red(红色)	Green(绿色)	Blue(蓝色)
Black(黑)	0	0	0
White(白)	1	1	1
Red(红)	1	0	0
Green(绿)	0	1	0
Blue(蓝)	0	0	1
Yellow(黄)	1	1	0
Magenta(品红)	1	0	1
Cyan(青)	0	1	1
Gray(灰)	0.5	0.5	0.5
Dark red(暗红)	0.5	0	0
Copper(铜色)	1	0.62	0.4
Aquamarine(碧绿)	0.49	1	0.83

【例 4.16】查看默认的色图矩阵。

```
peaks;            %以默认颜色显示 peaks 函数曲面
colormap
size(colormap)

ans =
    64     3
```

程序分析：peaks 函数的颜色如前图 4.15 所示，colormap 是 64×3 的矩阵，为了节省篇幅在此省略了中间的一些行数，每行为 RGB 颜色的相对亮度。第一行的颜色设定该曲面的最高点，最后一行的颜色设定该曲面的最低点，其余高度的颜色则根据线性内插法来决定。

(2) 预定义色图函数

表 4.8　预定义色图的函数表

命令	说明
hsv	HSV 的颜色对照表(默认值)，以红色开始和结束
hot	代表暖色对照表，黑、红、黄、白浓淡色
cool	代表冷色对照表，青、品红浓淡色
summer	代表夏天色对照表，绿、黄浓淡色
gray	代表灰色对照表，灰色线性浓淡色
copper	代表铜色对照表，铜色线性浓淡色
autumn	代表秋天颜色对照表，红、黄浓淡色
winter	代表冬天色对照表，蓝、绿浓淡色
spring	代表春天色对照表，青、黄浓淡色
bone	代表"X 光片"的颜色对照表
pink	代表粉红色对照表，粉红色线性浓淡色
flag	代表"旗帜"的颜色对照表，红、白、蓝、黑交错色
jet	HSV 的变形，以蓝色开始和结束
prim	代表三棱镜对照表，红、橘黄、黄、绿、蓝交错色

上表每行的函数默认产生一个 64×3 的色图矩阵，可以改变函数的参数产生一个 m×3 的色图矩阵。

【例 4.16 续】查看暖色色图。

```
colormap hot(8)           %产生暖色 peaks 函数曲面
colormap

ans =
    0.3333         0         0
    0.6667         0         0
    1.0000         0         0
    1.0000    0.3333         0
    1.0000    0.6667         0
    1.0000    1.0000         0
    1.0000    1.0000    0.5000
    1.0000    1.0000    1.0000
```

程序分析：hot(8)函数产生 8×3 的矩阵，表示黑、红、黄、白的浓淡色，在此图略，大家自己可以对比该图与前面图形的不同颜色。

2．色图的显示和处理

(1) 色图的显示

■ rgbplot 命令

语法：

rgbplot(map)

说明：map 是表 4.8 中的各预定义色图，rgbplot 命令可画出以行数为自变量红、绿、蓝相对亮度分量的直线图，反映 R、G、B 三色比重的变化。

■ colorbar 命令

colorbar 命令以不同颜色来代表曲面的高度，显示一个水平或垂直的颜色标尺。

【例 4.17】用 rgbplot 和 colorbar 命令显示色图，如图 4.21 所示。

```
subplot(2,1,1)
rgbplot(cool)          %画出冷色的颜色分量直线图
subplot(2,1,2)
peaks;
colormap cool          %产生冷色 peaks 函数曲面
colorbar               %显示颜色标尺
```

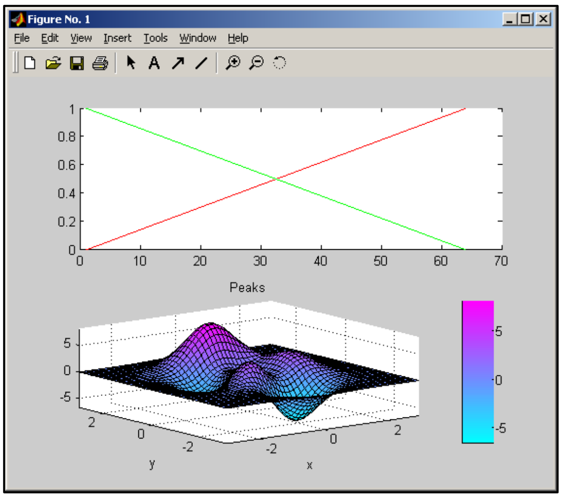


图 4.21 用 rgbplot 和 colorbar 命令显示色图

程序分析：rgbplot 画出红、绿、蓝三色分量，横坐标是 0～64 行，纵坐标是 0～1；colorbar 则显示高度与颜色的对照长条标尺，曲面上每一个小方块的颜色就是根据此对照图而得出的。

(2) 浓淡处理 shading

如果要使小片表面的颜色产生连续性的变化可使用 shading 命令，shading 命令的用法如表 4.9 所示。

表 4.9　shading 命令的用法

命令	功能
shading interp	使小片根据四顶点的颜色产生连续的变化，或根据网线的线段两端产生连续的变化，这种方式着色细腻但最费时。
shading flat	小片或整段网线的颜色是一种颜色。
shading faceted	在 flat 着色的基础上，同时在小片交接的边勾画黑色，这种方式立体表现力最强(默认方式)。

【例 4.18】使用浓淡处理 peaks 函数曲面图，如图 4.22 所示。

```
subplot(1,2,1)
peaks;
shading interp
subplot(1,2,2)
peaks;
shading faceted
```

(3) 亮度处理 brighten

【例 4.18 续】对 peaks 函数曲面加亮，并查看色图矩阵。

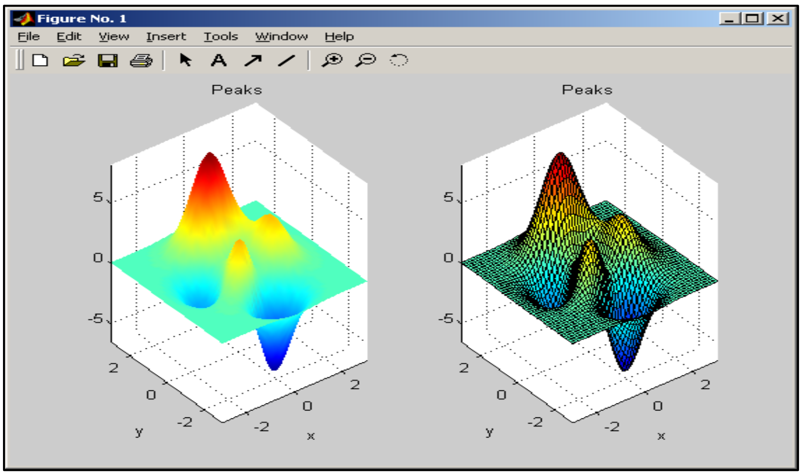


图 4.22 用 interp 和 faceted 方式进行浓淡处理

```
peaks;
brighten(0.5)
colormap
```

程序分析：可以通过图形查看亮度处理后的变化。

4.3 MATLAB 的特殊图形绘制

4.3.1 条形图

条形图常用于对统计的数据进行作图，特别适用于少量且离散的数据。绘制条形图的函数如表 4. 10 所示。

表 4. 10 条形图函数

函数	功能	函数	功能
bar	垂直条形图	bar3	三维垂直条形图
barh	水平条形图	bar3h	三维水平条形图

语法：

bar(x,y,width,'参数') %画条形图

bar3(y,z,width,'参数') %画三维条形图

说明：x 是横坐标向量，省略时默认值是 1:m，m 为 y 的向量长度；y 是纵坐标，可以是向量或矩阵，当是向量时每个元素对应一个竖条，当是 m×n 的矩阵时，将画出 m 组竖

条每组包含 n 条；width 是竖条的宽度，省略时默认宽度是 0.8，如果宽度大于 1，则条与条之间将重叠；'参数'有 grouped(分组式)和 stacked(累加式)，省略时默认为 grouped。bar3 命令的格式也相同，y 必须是单调增加或减小，省略时为 1:m；'参数'除了 grouped 和 stacked 还有 detached(分离式)。

【例 4.19】用条形图表示某年一月份中 3 日～6 日连续四天的温度数据，y 矩阵的各列分别表示平均温度、最高温度和最低温度，如图 4.23 所示，用条形图和三维条形图分别表示。

```
x=3:6;
y=[5.3000   13.0000    0.4000
5.1000   11.8000   -1.7000
3.7000    8.1000    0.6000
1.5000    7.7000   -4.5000]
bar(x,y)                 %画条形图
bar3(x,y)                %画三维条形图
```

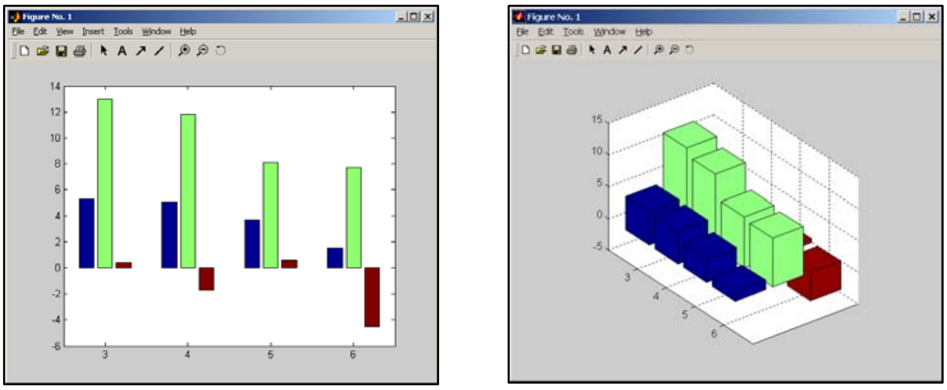


图 4.23 (a) 条形图　　　　(b) 三维条形图

程序分析：由上图看出条形图是按行分组的，每组为每天的平均温度、最高温度和最低温度。

4.3.2 面积图和实心图

1. 面积图

面积图是在曲线与横轴之间填充颜色，用于绘制面积图的命令为“area”，只能用于二维绘图。

语法：

area(y)　　　　　　%画面积图

area(x,y)

说明：y 可以是向量或矩阵，如果 y 是向量则绘制的曲线和 plot 命令相同，只是曲线和横轴之间填充颜色，如果 y 是矩阵则每列向量的数据构成面积叠加起来；x 是横坐标，当 x 省略时则横坐标为 1:size(y,1)。

2. 实心图

实心图是将数据的起点和终点连成多边形，并填充颜色，绘制实心图的命令为“fill”。

语法：

fill(x,y,c)　　　　　　%画实心图

说明：c 为实心图的颜色，可以用'r'、'g'、'b'、 'c'、'm'、'y'、'w'、'k'，或 RGB 三元组行向量表示，也可以省略。

【例 4.19 续】绘制面积图和实心图，并比较其区别，如图 4.24 所示。

```
area(x,y)           %面积图
fill(x,y,'r')          %红色的实心图
```

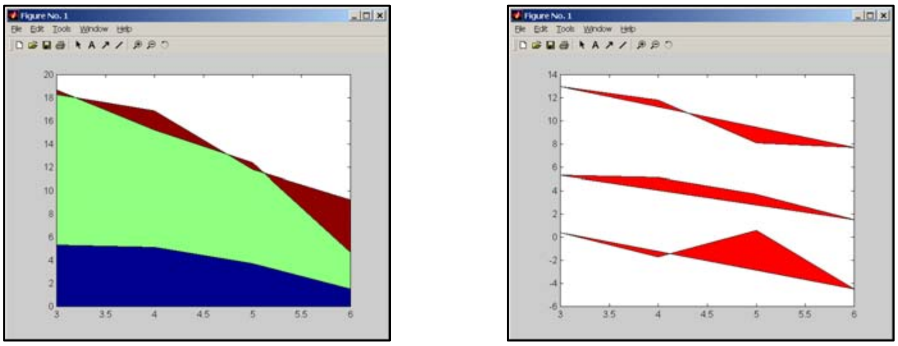


图 4.24 (a) 面积图　　　　**(b)** 实心图

程序分析：由上图可知面积图是绘制曲线和横轴间的面积，y 的各列叠加在一起的，而实心图是将起点和终点连接并填充颜色的多边形。

4.3.3 直方图

语法：

hist(y,m)　　　　　　%统计每段的元素个数并画出直方图

hist(y,x)

说明：m 是分段的个数，省略时则默认为 10；x 是向量，用于指定所分每个数据段的中间值；y 可以是向量或矩阵，如果是矩阵则按列分段。

【例 4.20】用直方图表示正态分布的随机数分布，如图 4.25 所示。

```
y=randn(10,2)          %产生 10*2 的正态分布的随机数矩阵

y =
   -1.1878   -1.1859
   -2.2023   -1.0559
    0.9863    1.4725
   -0.5186    0.0557
    0.3274   -1.2173
    0.2341   -0.0412
    0.0215   -1.1283
   -1.0039   -1.3493
   -0.9471   -0.2611
   -0.3744    0.9535
x=-2:0.5:2;
hist(y,x)
```

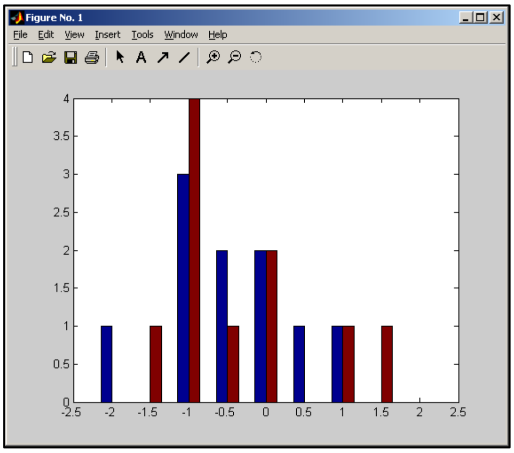


图 4.25 直方图

程序分析：直方图显示的是 y 在 x 附近的元素的个数，如-2 附近有一个。产生的随机数不同则得出的直方图也不同。

4.3.4 饼图

饼图是用于显示向量中的各元素占向量元素总和的百分比，可以用 pie 和 pie3 命令分别绘制二维和三维饼图。

语法：

pie(x,explode,’label’) %画二维饼图

pie3(x,explode,’label’) %画三维饼图

说明：x 是向量；explode 是与 x 同长度的向量，用来决定是否从饼图中分离对应的一部分块，非零元素表示该部分需要分离；’label’是用来标注饼图的字符串数组。

【例 4.21】绘制四个季度支出额的饼图，如图 4.26 所示。

```
y=[200 100 250 400];        %四个季度支出额
explode=[0 0 1 0];
pie(y,explode,{'第一季度','第二季度','第三季度','第四季度'})
```

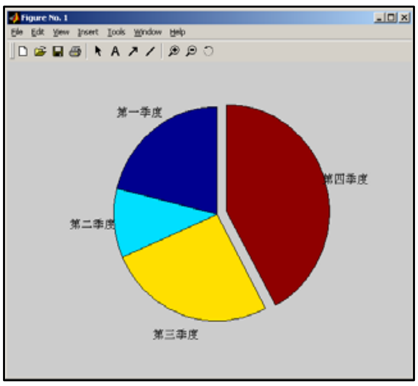


图 4.26 饼图

4.3.5 离散数据图

MATLAB 提供了多个绘制离散数据的命令，有 stem、stem3、stairs 和 scatter 等。

【例 4.22】使用几种绘制离散数据的命令来显示 $y = e^{-2x}\sin(x)$ 的离散数据，如图 4.27 所示。

```
x=0:0.1:2*pi;
y=sin(x).*exp(-2*x);
subplot(3,1,1)
stem(x,y,'filled')        %画火柴杆图
subplot(3,1,2)
stairs(x,y)               %画阶梯图
subplot(3,1,3)
scatter(x,y)              %画点图
```

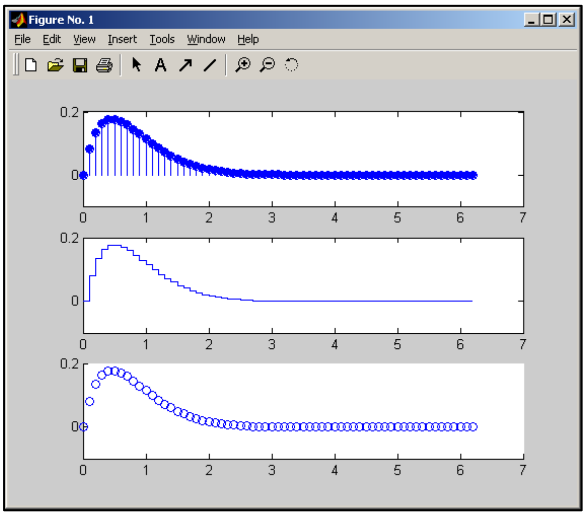


图 4.27 离散数据图

程序分析：'filled'参数是来填充火柴杆图的点标记。

4.3.6 对数坐标和极坐标图

1. 对数坐标图形

对数坐标图形有 semilogx、semilogy 和 loglog 命令。

语法：

semilogx(x,y,'参数') %绘制 x 为对数坐标的曲线

semilogy(x,y,'参数') %绘制 y 为对数坐标的曲线

loglog(x,y,'参数') %绘制 x、y 都为对数坐标的曲线

说明：参数和 plot 命令一样，只是坐标不同。

【例 4.23】求传递函数为 $G(s)=\dfrac{1}{s(0.5s+1)}$ 的对数幅频特性曲线，如图 4.28 所示，横坐标为 w 按对数坐标。

```
w=logspace(-2,3,20);           %频率 w 为 0.01 到 1000
Aw=1./(w.*sqrt((0.5*w).^2+1));    %计算幅频
```

```
Lw=20*log10(Aw);              %计算对数幅频
semilogx(w,Lw)
title('对数幅频特性曲线')
```

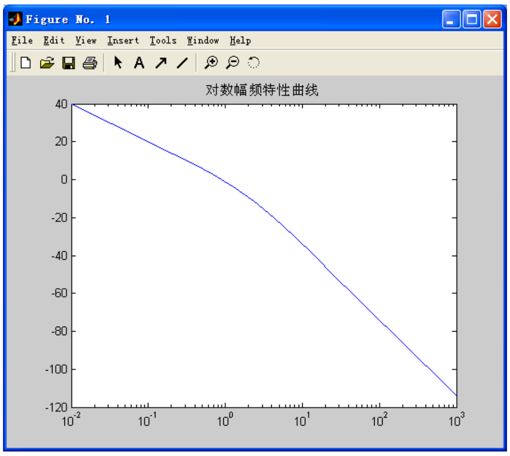


图 4.28 对数频率特性

2. 极坐标图

极坐标图由 polar 命令来实现。

语法：

polar(theta,radius,'参数')　　　　%绘制极坐标图

说明：theta 为相角，radius 为离原点的距离。

【例 4.23 续】用极坐标图表示上述传递函数的 Nyquist 曲线，如图 4.29 所示。

```
w=logspace(-2,3,20);
Fw=-90-atan(0.5*w);
polar(Fw,Aw)
```

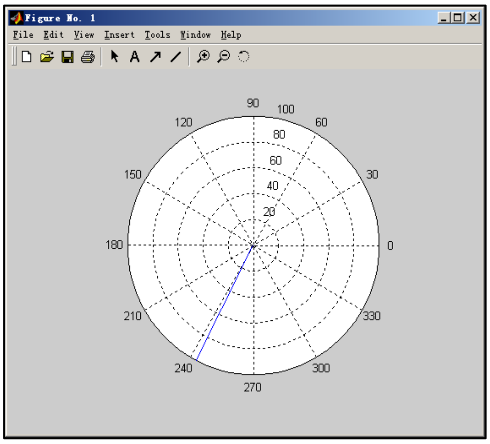


图 4.29 极坐标图

4.3.7 等高线图

语法：

contour(Z,n)　　　　　%绘制 Z 矩阵的等高线

contour(x,y,z,n)　　　%绘制以 x 和 y 指定 x、y 坐标的等高线

说明：n 为等高线的条数，省略时为自动条数。

【例 4.24】绘制 peaks 函数的等高线，如图 4.30 所示。

```
[x,y,z]=peaks;
contour(x,y,z)          %画二维等高线
contour3(z,30)          %画 30 条三维等高线
```

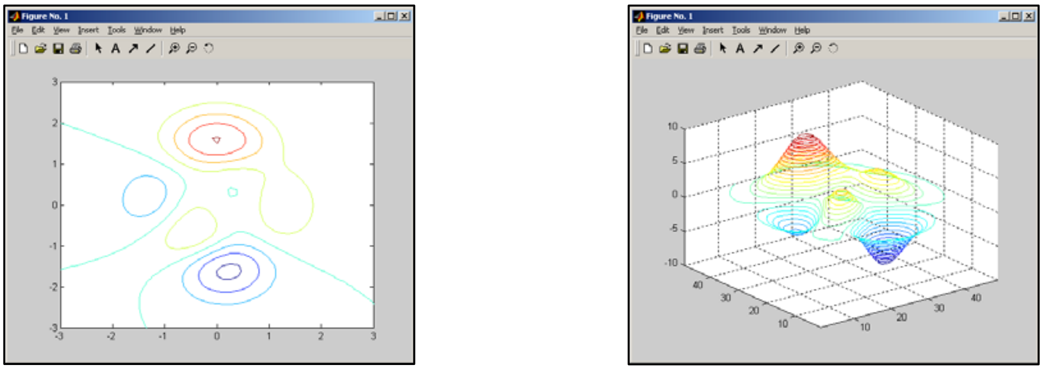


图 4.30 (a) 二维等高线　　(b) 三维等高线

4.3.8 复向量图

1. compass 命令

compass 绘制的是以原点为起点的一组复向量，因此又称为罗盘图。

语法：

compass(u,v)　　　%画罗盘图

compass(Z)

说明：u、v 分别为复向量的实部和虚部；当只有一个参数 Z 时，则相当于 compass(real(Z),imag(Z))。

2. feather 命令

feather 绘制的是起点为(k,0)的复向量图，又称为羽毛图。

语法：

feather(u,v)　　　%画羽毛图

feather (Z)

【例 4.25】用罗盘图和羽毛图绘制复向量，如图 4.31 所示。

```
theta=0:0.2:2*pi;
z=sin(theta).*exp(j*theta);
compass(z)
feather(z)
```

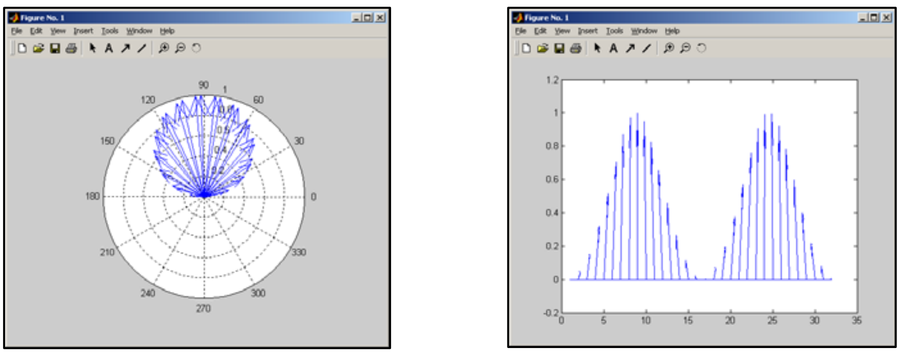


图 4.31 (a)罗盘图　　(b)羽毛图

程序分析：羽毛图的绘制起点是(k,0)，k 从 1～n，n 是 Z 向量的元素序号。

4.4 图形窗口的功能

1．工具栏

从 MATLAB 的图形窗口带有工具栏，工具栏如图 4.32 所示。

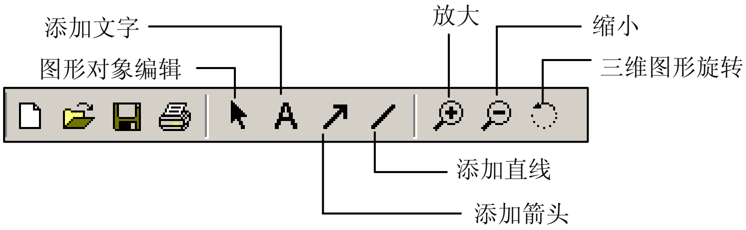


图 4.32 图形窗口的工具条

2．菜单

图形窗口中的 Edit 和 Insert 菜单可以方便地编辑图形，Edit 和 Insert 菜单如图 4.33 所示。

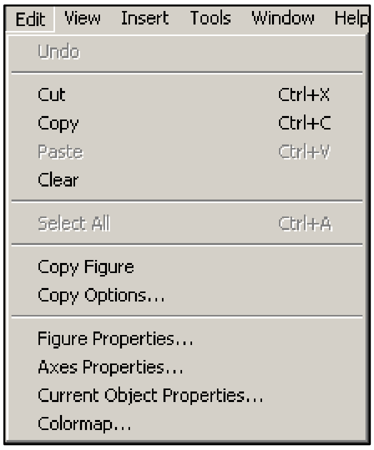


图 4.33 (a) Edit 菜单



(b) Insert 菜单

Edit 菜单：选择“Figure Properties…”、“Axes Properties…”和“Current Object Properties…”菜单项，可以打开相应的窗口来修改图形属性、坐标轴属性和对象属性。例如图 4.34 所示的坐标轴属性窗口，可以方便地设置坐标轴尺寸、类型、文字标注、视角等。

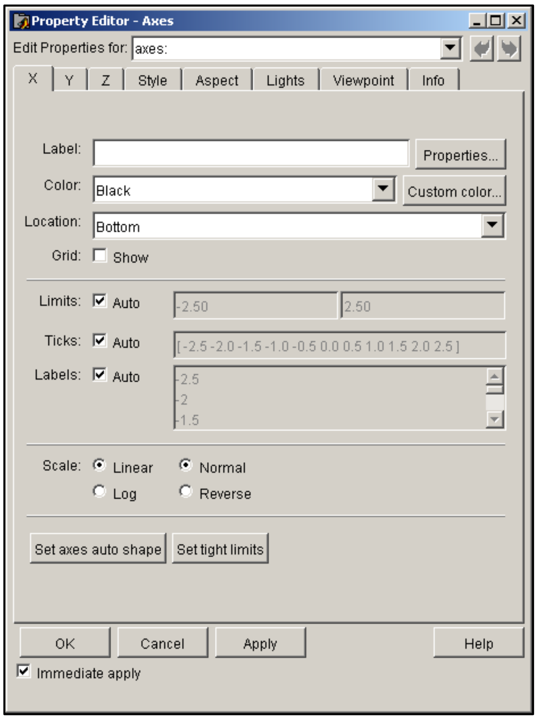


图 **4.34** 坐标轴属性窗口

Insert 菜单：可以插入各种文字标注、箭头、坐标轴等。

4.5 对话框

1. 输入信息对话框

输入对话框为用户的输入信息提供了界面，使用 inputdlg 命令创建。

语法：

answer = inputdlg(prompt,title,lineno,defans,addopts) %创建输入对话框

说明：answer 返回用户的输入，为元胞数组；prompt 为提示信息字符串，用引号括起来，为元胞数组；title 为标题字符串，用引号括起来，可以省略；lineno 用于指定输入值的行数，可以省略；defans 为输入项的默认值，用引号括起来，是元胞数组可以省略；addopts 指定对话框是否可以改变大小，取 on 或 off，省略时为 off 表示不能改变大小，为有模式对话框(有模式对话框是指在对话框关闭之前，用户无法进行其它程序的运行)，如果为 on 则可以改变大小，自动变为无模式对话框。

【例 4.26】利用输入对话框输入二阶系统的系数，如图 4.35 所示。

```
prompt={'请输入阻尼系数','请输入无阻尼振荡频率'};
defans={'0.707','1'};
p=inputdlg(prompt,'输入参数',1,defans)
```

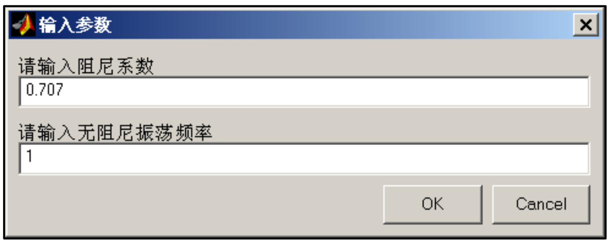


图 **4.35** 输入对话框

程序分析：prompt、defans 和 p 都是元胞数组。如果单击“Cancel”按钮，则返回空的元胞数组。

2. 输出信息对话框

语法：

msgbox(message,title,icon,icondata,iconcmap,CreateMode)　%创建消息框

说明：message 为显示的信息，可以是字符串或数组；title 为标题，是字符串可省略；icon 为显示的图标，可取值为“none”(无图标)、“error”(出错图标)、“help”(帮助图标)、“warn”(警告图标)或“custom”(自定义图标)，也可省略；当使用“custom”时，用 icondata 定义图标的数据，用 iconcmap 定义图标的颜色映象；CreateMode 为对话框的产生模式可省略，取值为“modal”(有模式)、“replace”(无模式可代替同名的对话框)、“non-modal”(默认为无模式)。

【例 4.26 续】使用消息框显示当阻尼系数大于 1 时的警告信息，如图 4.36 所示。

```
msgbox('阻尼系数输入范围出错','警告','warn')
```

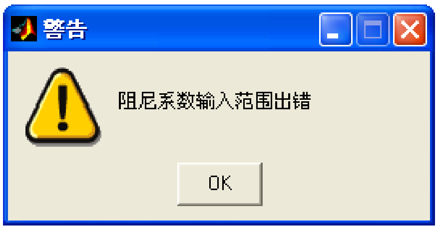


图 4.36 消息框

程序分析：消息框 msgbox 没有返回值。

【例 4.26 续】使用警告对话框显示当阻尼系数大于 1 时的警告信息。

```
warndlg('阻尼系数输入范围出错','警告')
```

程序分析：产生的对话框与图 4.36 一样。

【例 4.26 续】使用出错提示框显示当阻尼系数小于 0 时出错信息，如图 4.37 所示。

```
errordlg('阻尼系数输入出错','出错')
```

【例 4.26 续】使用帮助提示框显示阻尼系数的范围，如图 4.38 所示。

```
helpdlg('欠阻尼系数应大于 0 小于 1','帮助')
```

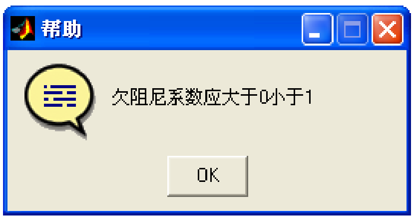


图 4.38 帮助提示框

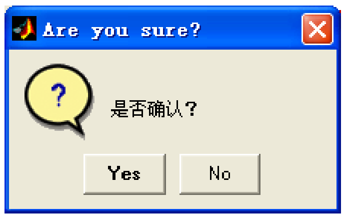


图 4.39 提问提

【例 4.26 续】使用提问提示框使用户确认，如图 4.39 所示。

```
button=questdlg('是否确认？','Are you sure?','Yes','No','Yes')
```

程序分析：有两个按钮，默认的指定按钮为“Yes”，即当用户按下回车键时，返回“Yes”，当用户单击按钮“No”时，返回“No”。

3. 文件管理对话框

(1) 打开文件对话框 uigetfile

uigetfile 命令提供了打开文件对话框，可以选择文件类型和路径。

语法：

[FileName, PathName] = uigetfile(FiltrEspec, Title,x,y)

说明：FileName 和 PathName 分别为返回的文件名和路径，可省略，如果按“取消”按钮或发生错误，都返回 0；FiltrEspec 指定初始时显示的文件名，可以用通配符“*”表示，当省略时，则自动列出当前路径下的所有“*.m”文件和目录；Title 为对话框标题，可省略；x、y 分别指定对话框在屏幕上的位置(到屏幕左上角的距离)，单位是像素，可省略。

【**例 4.27**】利用打开文件对话框选择 MATLAB 目录下的文件 license.txt，如图 4.40 所示。

```
[fname,pname]=uigetfile('*.*','打开文件',100,100)
```

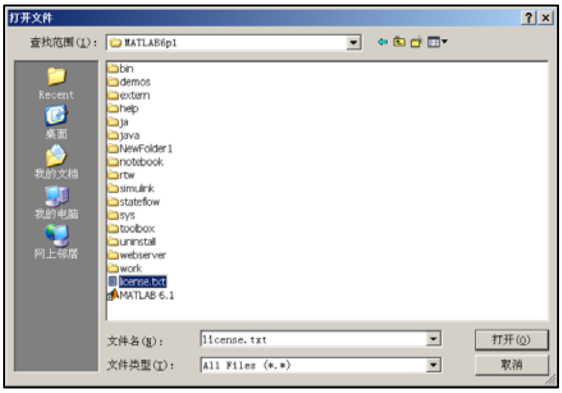


图 **4.40** 打开文件对话框

程序分析：在屏幕的(100，100)位置显示打开文件对话框，单击“打开”按钮，返回文件名和路径名到 fname 和 pname 变量。

(2) 保存文件对话框 uiputfile

uiputfile 命令提供了保存文件对话框，用来选择文件类型和路径。

语法：

[FileName, PathName] = uiputfile(FiltrEspec, Title,x,y)

说明：参数定义与 uigetfile 相同。

【**例 4.27 续**】利用保存文件对话框来选择文件。

```
[fname1,pname1]=uiputfile('Ex0431.mat','保存文件')
```

运行该命令则会出现保存文件对话框，如果要保存文件则在该语句后，添加第八章介绍的文件输入输出命令来实现。

4.6 句柄图形

4.6.1 句柄图形体系

句柄图形是一种面向对象的绘图系统，又称为低层图形。

句柄图形体系由若干个图形对象组成，如图 4.41 所示。

4.6.2 图形对象的操作

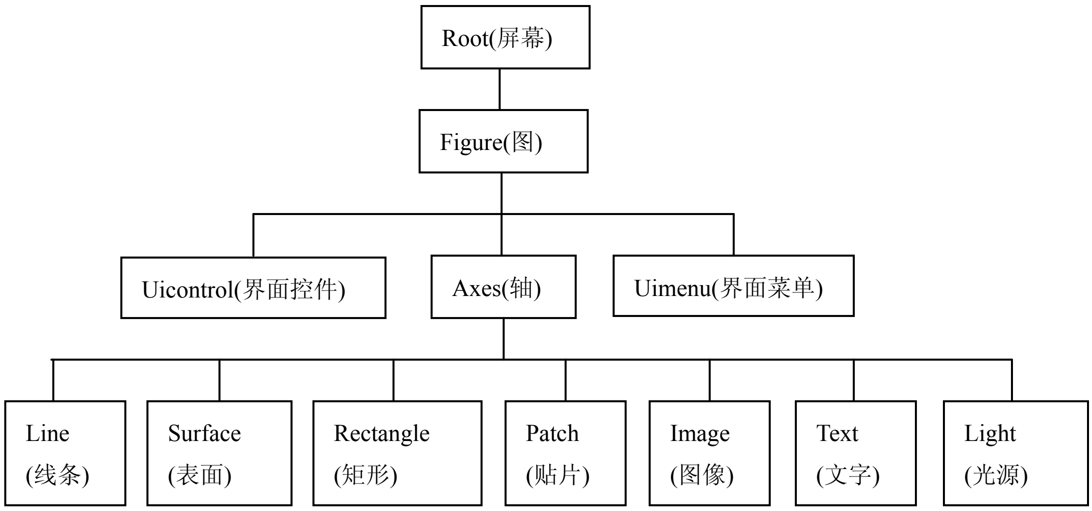


图 4.41 句柄图形体系

1．图形对象的创建

每次创建一个对象时，就为它建立一个唯一的句柄。每个命令的格式及功能如表 4.11 所示。

表 4.11 创建图形对象的命令

命令	功能	说明
h_ figure =figure(n)	创建第 n 个图形窗口	n 为正整数
h_axes =axes('position',[left,bottom,width,height])	创建坐标轴	定义轴的位置和大小
h_ line =line(x,y,z)	创建直线	z 省略则在二维平面上
h_ surface=surface(x,y,z,c)	创建面	x、y、z 定义三维曲面，c 是颜色参数
h_ rectangle= rectangle ('position',[x,y,w,h],'curvature',[xc,yc])	创建矩形	x、y 为左下顶点坐标，w、h 为长方形的宽和高，xc、yc 为曲率
h_ patch=patch('faces',fac,'veitices',vert)	创建贴片	fac 为多边形顶点的序号矩阵，vert 为顶点矩阵
h_ image=image(x)	创建图像	x 为图像数据矩阵
h_ text=text(x,y,'string')	创建文字	x、y 为字符串 string 的标注位置
h_ light=light('PropertyName',Propertyvalue)	创建光源	设置光的入射方向
h_ uicontrol =uicontrol('PropertyName',Propertyvalue)	创建用户界面控件	PropertyName 和 Propertyvalue 指定控件的类型
h_uimenu = uimenu ('propertyName', Propertyvalue)	创建用户界面菜单	propertyName 和 Propertyvalue 指定菜单的形式

2．对象句柄的获取

(1) 当前对象句柄的获取

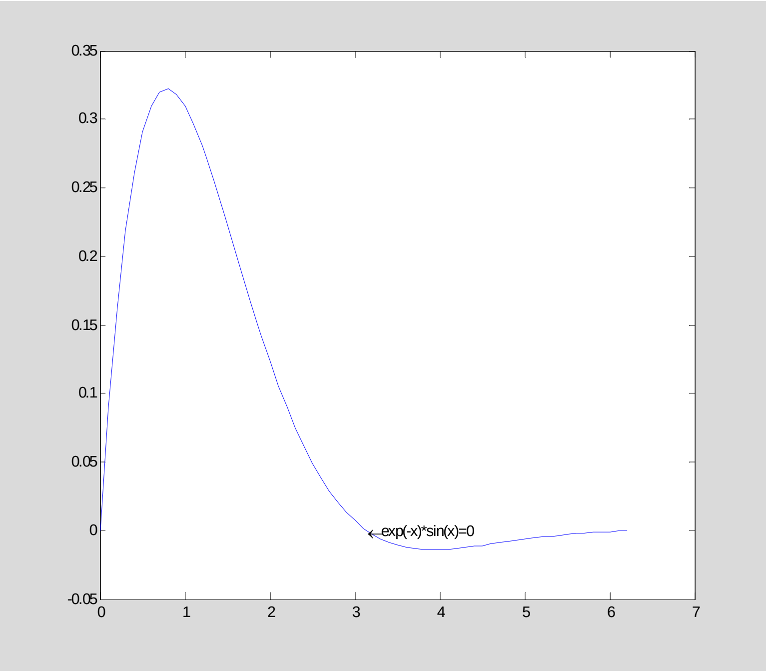
MATLAB 提供了三个获取当前对象句柄的命令，分别是 gcf、gca、gco。

语法：

```
gcf          %获取当前图形窗口句柄
gca          %获取当前坐标轴句柄
gco          %获取被鼠标最近点击对象的句柄
```

【例 4.28】使用命令获取图形对象的句柄，如图所示。

```
x=0:0.1:2*pi;
y=sin(x).*exp(-x);
plot(x,y)
text(pi,0,'\leftarrowexp(-x)*sin(x)=0')
h_fig=gcf          %获取图形窗口的句柄
```




```
h_fig =
     1

h_axes=gca    %获取坐标轴的句柄

h_axes =
  101.0013
h_obj=gco          %获取最近点击对象的句柄

h_obj =
     3.0017
```

程序分析：用 plot(x,y)画线，因为没有图形窗口所以自动产生图形窗口；图形窗口句柄为 1；坐标轴句柄为 100.0013；鼠标最近点击的对象是曲线句柄为 3.0017。

(2) 查找对象

用命令 findobj 可以快速查找所有对象，以及获取指定属性值的对象句柄。

语法：

h=findobj %返回根对象和所有子对象的句柄

h=findobj(h_obj) %返回指定对象的句柄

h=findobj('PropertyName',PropertyValue) %返回符合指定属性值的对象句柄

h=findobj(h_obj, 'PropertyName', PropertyValue)

%在指定对象及子对象中查找符合指定属性值的对象句柄

说明：h_obj 为指定对象句柄；PropertyName 为属性名；PropertyValue 为属性值。

【例 4.28 续】使用 findobj 命令获取图 4.42 中图形对象的句柄。

```
findobj                          %返回根对象和所有子对象的句柄

ans =
          0
     1.0000
   100.0013
   101.0038
     3.0017
```

```
h_text=findobj(h_fig,'string','\leftarrowexp(-x)*sin(x)=0') %查找符合属性值的文字对象句柄

h_text =
   101.0038
```

程序分析：根对象句柄为 0；其子对象图形窗口句柄为 1；图形窗口子对象坐标轴句柄为 100.0013；坐标轴子对象文字句柄为 101.0038；坐标轴子对象曲线句柄为 3.0017；文字对象的文字属性名为 string。

(3) 追溯父对象和子对象的句柄

如果一个对象的句柄已知，则可以追溯到其父对象和子对象的句柄。

语法：

h_parent=get(h_obj,'parent') %追溯父对象的句柄

h_children=get(h_obj,'children') %追溯子对象的句柄

【例 4.28 续】追溯坐标轴对象的父对象和子对象。

```
h_children=get(h_axes,'children')    %子对象为文字对象和曲线对象

h_children =
       101.0038
     3.0017

h_parent=get(h_axes,'parent')          %父对象为图形窗口对象

h_parent =
     1
```

3．对象句柄的删除

删除图形对象使用命令 delete(h_obj)，该命令将删除句柄所指对象和所有子对象，而且不提示确认，使用时要小心。

【例 4.28 续】删除坐标轴。

```
delete(h_axes)
```

4.6.3 图形对象属性的获取和设置

1．创建对象时设置属性

对象的属性可以在创建时设置，在创建时句柄图形对象可以设置多个属性。

【例 4.29】创建图形对象。

```
h_fig=figure('color','red','menubar','none','position',[0,0,300,300])
```

或者使用结构数组创建图形对象：

```
ps.color='red';
ps.position=[0,0,300,300];
ps.menubar='none';
h_fig=figure(ps)

h_fig =
     1
```

程序分析：创建一个窗口，背景为红色，没有菜单条，在屏幕的(0,0)位置，宽度、高度为 300。

2. 用 get 函数获取属性值

get 函数用于获取指定对象的属性值。

语法：

get(h_obj)	%获取句柄对象所有属性的当前值
get(h_obj, 'PropertyName')	%获取句柄对象指定属性的当前值

【例 4.29 续】获取图形对象属性。

```
p=get(h_fig,'position')

p =
     0     0   300   300
c=get(h_fig,'color')

c =
     1     0     0
```

程序分析：图形对象的颜色为红色，用 RGB 三元组表示。

3. 用 set 函数设置属性值

set 函数用来设置对象的属性值。

语法：

set(h_obj)	%显示句柄对象所有属性和属性值
set(h_obj, 'PropertyName')	%显示句柄对象指定属性名的属性值

set(h_obj, 'PropertyName', ' PropertyValue ') %设置句柄对象指定属性的属性值

set(h_obj, 'PropertyStructure') %用结构数组设置句柄对象指定属性的属性值

【例 4.30】使用低层命令画图，并设置各对象的属性，如图 4.43 所示。

```
h_fig=figure('color','red','menubar','none','position',[0,0,300,300])
;
    x=0:0.1:2*pi;
    y=sin(x).*exp(-x);
    h_line1=plot(x,y,'b');
    title('y=exp(-x)*sin(x)')
    set(gca,'ygrid','on')                              %显示 y 网格
   line1width=get(h_line1,'linewidth')                  %获取曲线宽
度

   line1width =
      0.5000


    set(h_line1,'linewidth',3)                          %设置曲线宽
度
    h_title =get(gca,'title')                          %获取标题句柄

   h_title =
     115.0016


    titlefontsize=get(h_title_fontsize,'fontsize')                  %获
取字体大小
    set(h_title_fontsize,'fontsize',13)                       %设置标
题字体大小
    h_text1=text(pi,0,'\downarrow');                   %画向下箭头
    text1pos=get(h_text1,'position')                     %获取文字位
置

h_text2=text(text1pos(1,1),text1pos(1,2)+0.025,'exp(-x)*sin(x)=0');
   %设置文字位置
    set(h_text1,'fontsize',13,'color','red')                       %设
置字体大小、颜色
    set(h_text2,'fontsize',13,'color','red')
```

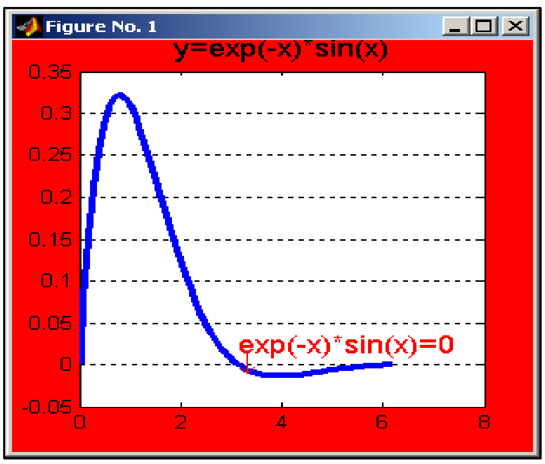


图 **4.43** 图形对象

4．对象属性的默认设置和获取

对象属性的默认值也可以设置和获取。

语法：

get(h_obj, 'DefaultObjectTypePropertyName')　　　　%获取对象属性的默认值

set(h_obj, 'DefaultObjectTypePropertyName', PropertyValue)

%设置属性的用户定义默认值

set(h_obj, 'DefaultObjectTypePropertyName', 'Remove')

%删除属性的用户定义默认值

说明：DefaultObjectTypePropertyName 的表示为“Default+对象名+属性名”，例如线对象的线条宽度为“DefaultLineLineWidth”。

4.7 图形用户界面(GUI)设计

MATLAB 设计图形用户界面的方法有两种：使用可视化的界面环境和通过编写程序。

4.7.1 可视化的界面环境

MATLAB 提供了可视化的界面环境 Guide 打开可视化界面环境的方法有以下几种：

(1) 选择菜单“File”——“ New”——“GUI”命令；

(2) 在命令窗口输入“Guide”命令或输入“Guide Filename”就会出现 Guide 快速开始界面。如图 4.44 所示。

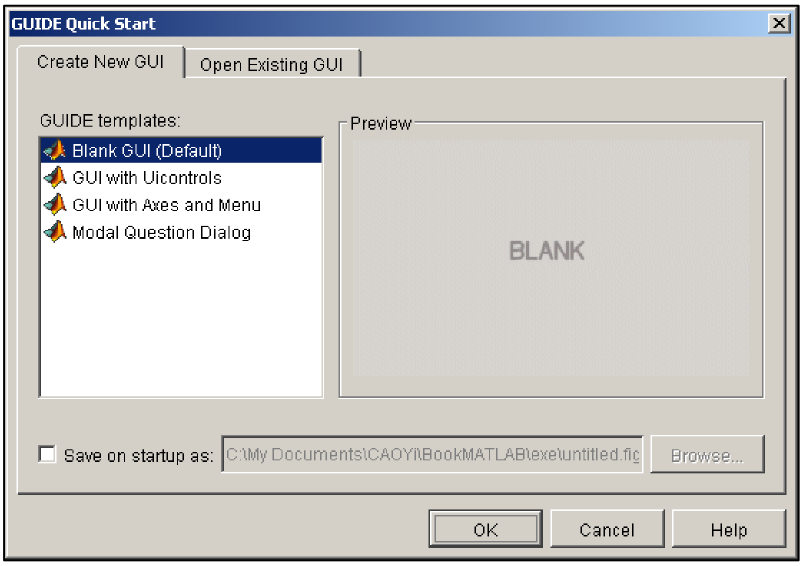


图 4.44 Guide 快速开始界面

在 Guide 快速开始界面中有“Create New GUI”和“Open Existing GUI”两个选项卡，选择“Blank GUI(Default)”，然后单击“OK”按钮，就会出现空白的可视化界面窗口，如图 4.45 所示。如果需要创建具有控件或坐标轴、菜单等的界面，可以单击图 4.44“Blank GUI(Default)”下面的“GUI with Uicontrols”等选项。

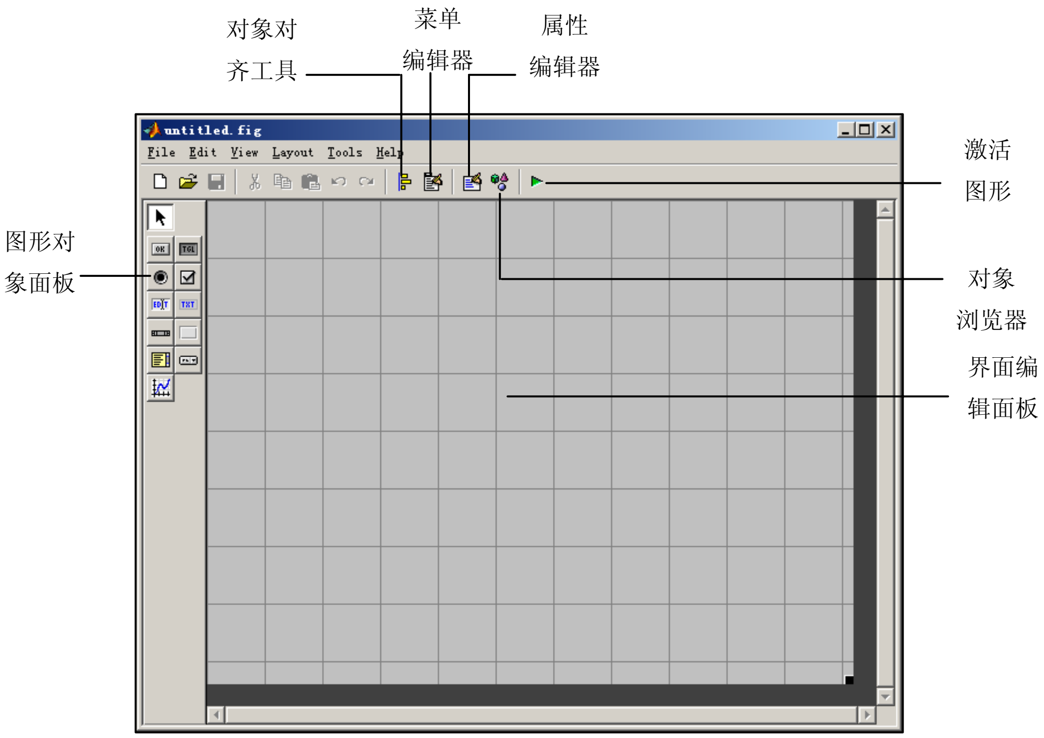


图 4.45 可视化界面环境

可视化界面环境如图在工具栏主要提供了四个工具：对象对齐工具(Align Objects)、菜单编辑器(Menu Editor)、属性编辑器(Property Inspector)和对象浏览器(Object Browser)，单击这四个按钮就会出现相应的窗口。在可视化的界面环境的左边是图形对象面板，有各种控件可以通过拖放到空白的界面编辑面板来创建新控件。

4.7.2 菜单

1. 菜单编辑器

【例 4.31】使用菜单编辑器创建菜单。

在菜单编辑器创建菜单，如图 4.47 所示，如果是直接在可视化的界面环境中新建图形窗口，则从头开始新建菜单，如图 4.47(a)；如果在已存在的图形窗口中创建菜单，则 MATLAB 图形窗口默认有七个标准菜单，新建的菜单从最右边添加，如图 4.47(b)。

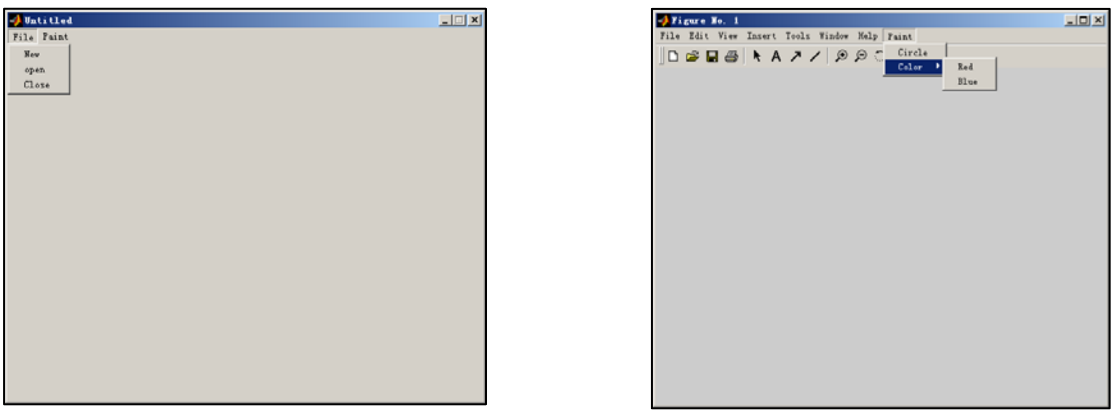


图 4.47 (a)在新窗口创建菜单 (b)在已建

在图 4.47(a)中创建了两个菜单“File”和“Paint”，在图 4.47(b)中创建了一个菜单“Paint”，并具有两级下拉子菜单，“Circle”和“Color”为第一级下拉菜单，“Red”和“Blue”为第二级。

2. 编程创建菜单

【例 4.31 续】编程创建如上图 4.47(b)所示的菜单。

```
h_fig=gcf
h_fig =
      1
h_menu=uimenu(h_fig,'label','Paint');              %创建菜单 Paint
h_menu1=uimenu(h_menu,'label','Circle');           %创建 Paint 的子菜单 Circle
h_menu2=uimenu(h_menu,'label','Color');            %创建 Paint 的子菜单 Color
h_menu21=uimenu(h_menu2,'label','Red');        %创建 Color 的子菜单 Red
h_menu22=uimenu(h_menu2,'label','Blue');       %创建 Color 的子菜单 Blue
```

程序分析：label 属性用来命名菜单项名称，和图 4.46 的菜单编辑器中的 label 栏一致。

3. 回调函数

【例 4.31 续】将已创建的菜单修改，并添加回调函数。

```
h_menu21=uimenu(h_menu2,'label','Red','callback','set(h_fig,''color'',''red'')')
%创建 Color 的子菜单 Red 将图形背景为红色
h_menu22=uimenu(h_menu2,'label','Blue','callback','set(h_fig,''color'',''blue'')')
%创建 Color 的子菜单 Blue 将图形背景为蓝色
```

4.7.3 控件

1. 常用控件

常用控件的作用如表 4.12 所示。

表 4.12 控件的功能

控件名	PropertyName	功能
按钮	PushButton	最常用的控件，用于响应用户的鼠标单击，按钮上有说明文字说明其作用。
切换按钮	ToggleButton	当单击时会凹凸状态切换。
单选按钮	RadioButton	当单击时会用黑白点切换，总是成组出现，多个单选按钮互斥，一组中只有一个被选中。
复选框	CheckBox	当单击时会用 √ 切换，有选中、不选中和不确定等状态，总是成组出现，多个复选框可同时选用。
文本框	EditText	凹形方框，可随意输入和编辑单行和多行文字，并显示出来。
静态文本框	StaticText	用于显示文字信息，但不接受输入。
滚动条	Slider	可以用图示的方式显示在一个范围内数值的大概值范围，用户可以移动滚动条改变数值。
框架	Frame	将一组控件围在框架中，用于装饰界面。
列表框	ListBox	显示下拉文字列表，用户可以从列表中选择一项和多项。
弹出式菜单	PopupMenu	相当于文本框和列表框的组合，用户可以从下拉列表中选择。
坐标轴	Axes	用于绘制坐标轴。

2. 控件的创建

(1) 在可视化界面环境中创建控件

在可视化界面环境中创建控件很简单，就是在图形对象面板中选中控件，然后在空白的界面编辑面板中拖放即可，如图 4.48 所示为各种控件的显示。

(2) 用 unicontrol 命令创建控件

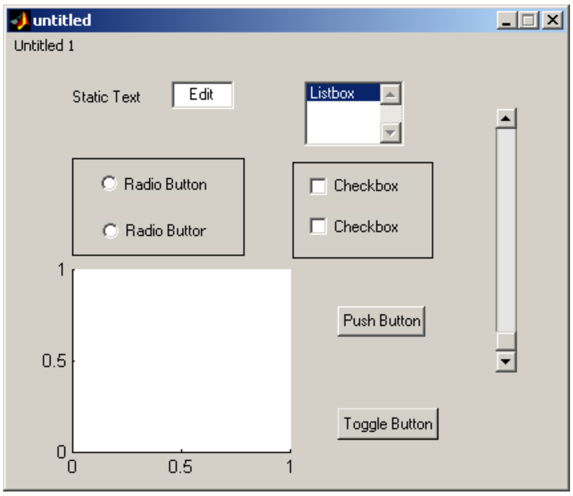


图 4.48 可视化的界面环境

语法：

h_control=uicontrol(h_Parent,’PropertyName’,ProperValue,…)

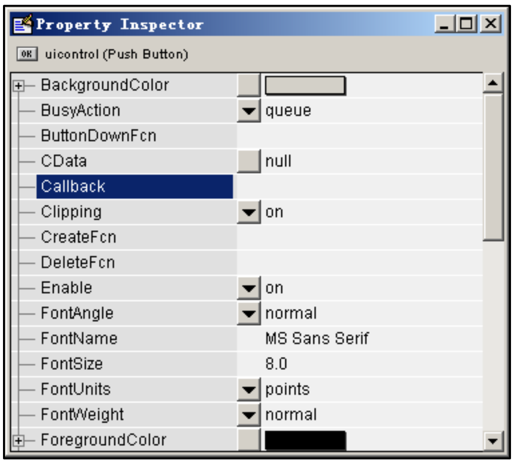


图 4.50 属性编辑器

4.7.5 回调函数

4.7.6 GUI 应用举例

【例 4.32】使用控件设计用户界面，根据阻尼系数绘制二阶系统的时域曲线。

功能：在图形用户界面中，通过弹出式菜单选择二阶系统的阻尼系数，然后单击不同按钮在坐标轴中绘制不同阻尼系数不同颜色的时域曲线。

1. 设计界面

使用“guide”命令打开 Guide 快速开始界面，选择“Blank GUI”出现空白的可视化界面环境窗口，调整图形大小，将界面窗口右边的图形对象面板中的控件拖放到空白窗口中。

放置以下控件：一个坐标轴、两个静态文本框、一个弹出式菜单、两个按钮；然后打开对象对齐工具对齐各控件，界面布局如图 4.52 所示。

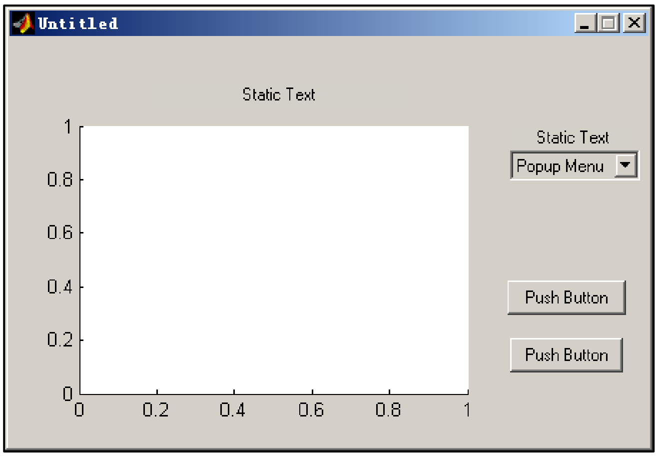


图 4.52 界面布局

2. 设置控件属性

各控件属性设置如表 4.13 所示。

表 4.13 各控件属性设置表

控件类型	属性名	属性值
静态文本框	String(显示文字)	二阶系统时域曲线
静态文本框	String(显示文字)	输入阻尼系数：
按钮	String(显示文字)	红色曲线
	Tag(标记)	pushbutton_red
按钮	String(显示文字)	蓝色曲线
	Tag(标记)	pushbutton_blue
弹出式菜单	String(显示文字)	0 0.3 0.5 0.707
	Tag(标记)	popupmenu_zeta
坐标轴	XLim(x 轴范围)	[0 20]
	YLim(y 轴范围)	[0 1.5]

单击工具栏的激活图形图标，或选择菜单“Tools”——“Activate Figure”命令，可以查看到运行的用户界面。

3. 回调函数

在设计界面中选定需要编写回调函数的按钮，然后选择菜单“View”——“Object Callbacks”——“Callback”，可以看到三个空白的 function，名字以控件的“Tag”属性取名，分别是“popupmenu_zeta”、“pushbutton_red”和“pushbutton_blue”，则出现相应的.m 文件如下：

```
% --------------------------------------------------------------------
function varargout = pushbutton_red_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = pushbutton_blue_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = popupmenu_zeta_Callback(h, eventdata, handles, varargin)
```

弹出式菜单的回调函数程序如下：

```
% --------------------------------------------------------------------
function varargout = popupmenu_zeta_Callback(h, eventdata, handles, varargin)
val=get(h,'value')
switch val
case 1
    handles.data=0
case 2
    handles.data=0.3
case 3
    handles.data=0.5
case 4
    handles.data=0.707
```

```
end
guidata(h,handles)
```

程序分析：h 为弹出式菜单的句柄，'Value'属性是弹出式菜单的所选值，根据用户在弹出式菜单的选择，确定阻尼系数；使用了 switch 结构，在下章中介绍，是用来作分支选择的；handles 变量非常重要，它是一个结构数组，包括两部分内容：

(1) 存储所有在图形界面中的控件、菜单、坐标轴对象的句柄，每个对象的句柄名称以对象的“Tag”名相同。

(2) 用于在 function 之间传递数据，先给 handles 结构数组建立一个域，然后调用 guidata 函数产生新的 handles 结构，并存储数据，以备其它 function 使用。

按钮的回调函数如下：

```
 % --------------------------------------------------------------------
function varargout = pushbutton_red_Callback(h, eventdata, handles, varargin)
x=0:0.1:20;
zeta=handles.data
y=1-1/sqrt(1-zeta^2)*exp(-zeta*x).*sin(sqrt(1-zeta^2)*x+acos(zeta));
plot(x,y,'r')

% --------------------------------------------------------------------
function varargout = pushbutton_blue_Callback(h, eventdata, handles, varargin)
x=0:0.1:20;
zeta=handles.data
y=1-1/sqrt(1-zeta^2)*exp(-zeta*x).*sin(sqrt(1-zeta^2)*x+acos(zeta));
plot(x,y,'b')
```

程序分析：两个按钮是用来画曲线的，zeta 变量存放阻尼系数，是从 handles 结构数组的 data 域获取的。

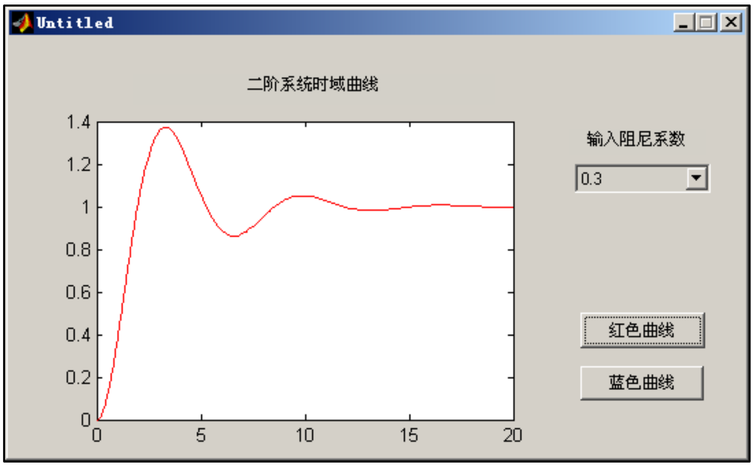


图 4.53 运行界面

运行界面如图 4.53 所示。

4.8 动画

4.8.1 以电影方式产生动画

以电影方式产生动画，有两个步骤：

(1) 使用 getframe 命令来抓取图形作为画面，每个画面都是以一个列向量的方式，置于存放整个电影的矩阵 M 中。

(2) 使用 movie(M,k)命令来播放电影，并可指定矩阵 M 播放的重复次数 k。

【例 4.33】使用电影方式制作动画，显示二阶系统的时域波形，最后一个画面如图 4.54 所示。

```
 n=20;
 for i=1:n
 x=0:0.1:i;
 y=1-1/sqrt(1-0.3^2)*exp(-0.3*x).*sin(sqrt(1-0.3^2)*x+acos(0.3));
 plot(x,y)
 axis([0,20,0,1.5]);                    %固定坐标轴
 M(i)=getframe;                     %抓取画面
end
 movie(M,3)                   %播放 3 次
```

程序分析：使用 for 循环，画 20 个不同阶段的波形画面，将画面抓取保存到 M 矩阵中，播放 3 次。

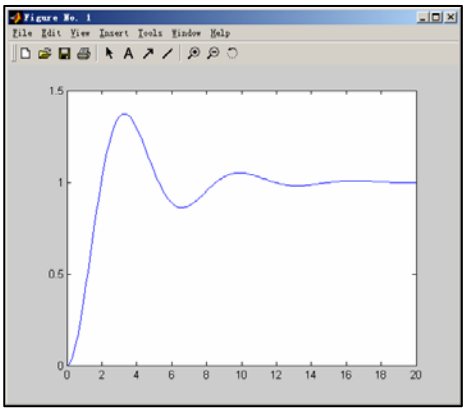


图 4.54 最后一帧画面

4.8.2 以对象方式产生动画

1．擦除属性 EraseMode

以对象方式产生动画需要设置 EraseMode 属性，EraseMode 为一个字符串，代表对象的擦除方式，即对于旧对象的处理方式。EraseMode 属性有以下几种：

- normal：计算整个画面的数据，重画整个图形。
- xor：将旧对象的点以 xor 的方式还原，即只画与屏幕色不一致的新对象点，擦除不一致的原对象点，这种方式不会擦除被擦对象下面的其他图像。
- background：将旧对象的点变成背景颜色，实现擦除，这种方式会擦除被擦对象下面

的其他图像。

- none：保留旧对象的点，不做任何擦除。

在上述四种 EraseMode 中，耗费时间的次序是：

normal > xor > background > none

2．对象的位置属性

通常在动画过程中，会改变对象的位置或尺寸、颜色等外观属性，位置属性有：

- xdata：为一个向量，代表对象的 x 坐标值。
- ydata：为一个向量，代表对象的 y 坐标值。

3．屏幕刷新

当新对象的属性设置后，应刷新屏幕，使新对象显示出来，刷新屏幕用 drawnow 命令实现。

4．产生动画

产生动画的具体步骤是：

(1) 先产生一个对象，其 EraseMode 属性为 xor、background 或 none；

(2) 在循环中产生动画，每次循环改变此对象的 xdata 或 ydata(或两者)；

(3) 使用 drawnow 命令刷新屏幕

【例 4.34】使用对象方式产生用一个红色的小球沿着曲线运动的动画，如图 4.55 所示。

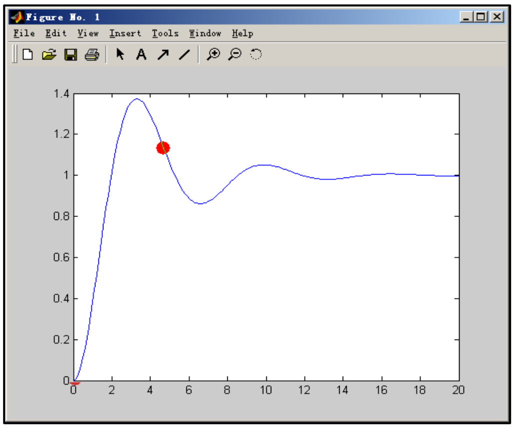


图 4.55 运行界面

```
    x=0:0.1:20;
    y=1-1/sqrt(1-0.3^2)*exp(-0.3*x).*sin(sqrt(1-0.3^2)*x+acos(0.3));
    plot(x,y)

h=line(0,0,'color','red','marker','.','markersize',40,'erasemode','xo
r');    %定义红色的小球
    for i=1:length(x)
       set(h,'xdata',x(i),'ydata',y(i));  %设置小球的新位置
       pause(0.005)                                %暂停 0.005 秒
       drawnow                                             %刷新屏幕
    end
```

程序分析：小球以 xor 的方式擦除旧曲线，如果将 EraseMode 改成 background 方式，则会发现小球会擦掉原来的曲线；drawnow 命令的作用是使 MATLAB 立刻处理 set 命令，

但由于本例中使用 pause 命令暂停，屏幕一定会得到及时的更新，drawnow 如果去掉效果一样。

第 5 章　MATLAB 程序设计

5.1 脚本文件和函数文件

M 文件有两种形式：M 脚本文件和 M 函数文件。

5.1.1 M 文本编辑器

MATLAB 的 M 文件是通过 M 文件编辑 / 调试器窗口(Editor / Debugger)来创建的。

单击 MATLAB 桌面上的图标，或者单击菜单“File”——“New”——“M-file”，可打开空白的 M 文件编辑器，也可以通过打开已有的 M 文件来打开 M 文件编辑器。如图 5.1 所示为打开已创建的 M 文件。

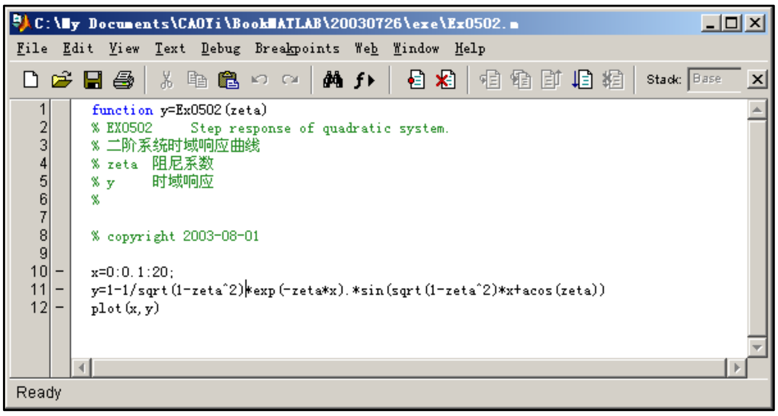


图 5.1 M 文件编辑/调试器窗口

5.1.2 M 文件的基本格式

下面介绍绘制二阶系统时域曲线的 M 文件，欠阻尼系统的时域输出 y 与 x 的关系为 $y=1-\frac{1}{\sqrt{1-\varsigma^2}}e^{-\varsigma x}\sin(\sqrt{1-\varsigma^2}x+a\cos\varsigma)$，【例 5.1】为 M 脚本文件，【例 5.2】为 M 函数文件。

【例 5.1】用 M 脚本文件绘制二阶系统时域曲线。

```
%EX0501    二阶系统时域曲线
%画阻尼系数为 0.3 的曲线
```

```
x=0:0.1:20;
y1=1-1/sqrt(1-0.3^2)*exp(-0.3*x).*sin(sqrt(1-0.3^2)*x+acos(0.3))
plot(x,y1,'r')
```

【例 5.2】创建一个画二阶系统时域曲线的函数，阻尼系数 zeta 为函数的输入参数。

```
function y=Ex0502(zeta)
% EX0502        Step response of quadratic system.
%  二阶系统时域响应曲线
% zeta    阻尼系数
% y         时域响应
%

% copyright 2003-08-01

x=0:0.1:20;
y=1-1/sqrt(1-zeta^2)*exp(-zeta*x).*sin(sqrt(1-zeta^2)*x+acos(zeta))
plot(x,y)
```

M 函数文件的基本格式：

函数声明行

H1 行(用%开头的注释行)

在线帮助文本(用%开头)

编写和修改记录(用%开头)

函数体

例如，在命令窗口输入 help 和 lookfor 命令查看帮助信息：

```
 help Ex0502
   EX0502        Step response of quadratic system.
   二阶系统时域响应曲线
   zeta    阻尼系数
   y         时域响应
 lookfor '二阶系统时域响应'
Ex0502.m: %二阶系统时域响应
```

5.1.3 M 脚本文件

脚本文件的特点：

(1) 脚本文件中的命令格式和前后位置，与在命令窗口中输入的没有任何区别。

(2) MATLAB 在运行脚本文件时，只是简单地按顺序从文件中读取一条条命令，送到 MATLAB 命令窗口中去执行。

(3) 与在命令窗口中直接运行命令一样，脚本文件运行产生的变量都是驻留在 MATLAB 的工作空间(workspace)中，可以很方便地查看变量，除非用 clear 命令清除；脚本文件的命令也可以访问工作空间的所有数据，因此要注意避免变量的覆盖而造成程序出错。

【例 5.1 续】在 M 文件编辑 / 调试器窗口中编写 M 脚本文件绘制二阶系统的多条时域曲线。

(1) 单击 MATLAB 桌面上的图标打开 M 文件编辑器。

(2) 将命令全部写入 M 文件编辑器中，为了能标志该文件的名称，在第一行写入包含文件名的注释。保存文件为 Ex0501.m。

```
%EX0501       二阶系统时域曲线
x=0:0.1:20;
y1=1-1/sqrt(1-0.3^2)*exp(-0.3*x).*sin(sqrt(1-0.3^2)*x+acos(0.3))
plot(x,y1,'r')                    %画阻尼系数为 0.3 的曲线
hold on
y2=1-1/sqrt(1-0.707^2)*exp(-0.707*x).*sin(sqrt(1-0.707^2)*x+acos(0.707))
plot(x,y2,'g')                    %画阻尼系数为 0.707 的曲线
y3=1-exp(-x).*(1+x)
plot(x,y3,'b')                    %画阻尼系数为 1 的曲线
```

(3) 选择 M 文件编辑器菜单“Debug”——“Run”，就可以在图形窗中看到如图 5.2 所示的曲线。

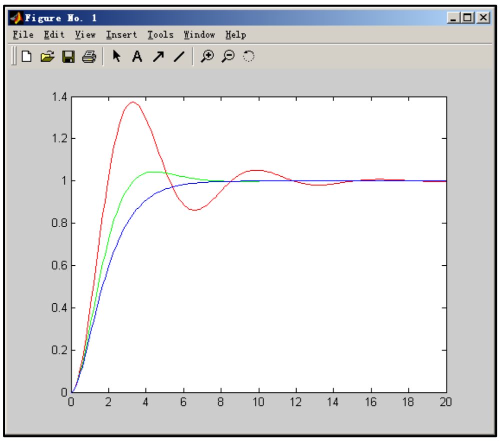


图 5.2 运行界面

查看工作空间的变量：

```
whos

  Name      Size                    Bytes  Class

  x         1x201                    1608  double array
  y1        1x201                    1608  double array
  y2        1x201                    1608  double array
  y3        1x201                    1608  double array

Grand total is 804 elements using 6432 bytes
```

5.1.4 M 函数文件

函数文件的特点：

(1) 第一行总是以“function”引导的函数声明行；

函数声明行的格式：

function [输出变量列表] = 函数名(输入变量列表)

(2) 函数文件在运行过程中产生的变量都存放在函数本身的工作空间；

(3) 当文件执行完最后一条命令或遇到“return”命令时，就结束函数文件的运行，同时函数工作空间的变量就被清除；

(4) 函数的工作空间随具体的 M 函数文件调用而产生，随调用结束而删除，是独立的、临时的，在 MATLAB 运行过程中可以产生任意多个临时的函数空间。

【例 5.2 续】在 M 文件编辑 / 调试器窗口编写计算二阶系统时域响应的 M 函数文件，并在 MATLAB 命令窗口中调用该文件。

创建 M 函数文件并调用的步骤如下：

(1) 编写函数代码

```
function y=Ex0502(zeta)
%EX0502        画二阶系统时域曲线
x=0:0.1:20;
y=1-1/sqrt(1-zeta^2)*exp(-zeta*x).*sin(sqrt(1-zeta^2)*x+acos(zeta))
plot(x,y)
```

(2) 将函数文件保存为“Ex0502.m”。

(3) 在 MATLAB 命令窗口输入以下命令，则会出现 f 的计算值和绘制的曲线：

```
f=Ex0502(0.3)
```

程序分析：

- 第一行指定该文件是函数文件，文件名为“Ex0502”，输入参数为阻尼系数 zeta，输出参数为时域响应 y。
- 当函数文件调用结束，查看 x、y：

```
 x
??? Undefined function or variable 'x'.
 y
??? Undefined function or variable 'y'.
```

注意：M 脚本文件和 M 函数文件的文件名及函数名的命名规则与 MATLAB 变量的命名规则相同。

5.2 程序流程控制

5.2.1 for ... end 循环结构

语法：

```
for  循环变量=array
      循环体
end
```

说明：循环体被循环执行，执行的次数就是 array 的列数，array 可以是向量也可以是矩阵，循环变量依次取 array 的各列，每取一次循环体执行一次。

【例 5.3】使用 for ... end 循环的 array 向量编程求出 1+3+5...+100 的值。

```
% EX0503        使用向量 for 循环
sum=0;
for n=1:2:100
    sum=sum+n;
end
sum
```

```
sum =
        2500
```

计算的结果为：sum =2500。

程序说明：循环变量为 n，n 对应为向量 1:2:100，循环次数为向量的列数，每次循环 n 取一个元素。

【例 5.4】使用 for ... end 循环的 array 矩阵编程将单位阵转换为列向量。

```
% EX0504    使用矩阵 for 循环
sum=zeros(6,1);
for n=eye(6,6)
    sum=sum+n;
end

    sum

sum =
     1
     1
     1
     1
     1
     1
```

程序分析：循环变量 n 对应为矩阵 eye(6,6)的每一列，即第一次 n 为[1;0;0;0;0;0]，第一次 n 为[0;1;0;0;0;0]；循环次数为矩阵的列数 6。

5.2.2 while ... end 循环结构

语法：

```
while 表达式
    循环体
end
```

说明：只要表达式为逻辑真，就执行循环体；一旦表达式为假，就结束循环。表达式可以是向量也可以是矩阵，如果表达式为矩阵则当所有的元素都为真才执行循环体，如果表达式为 nan，MATLAB 认为是假，不执行循环体。

【例 5.5】与【例 5.3】相同，计算 1+3+5...+100 的值。

```
% EX0505    使用 while 循环
sum=0;
n=1;
while n<=100
    sum=sum+n;
    n=n+2 ;
end

sum

n

sum =
```

```
        2500
n =
   101
```

程序分析：可以看出 while ... end 循环的循环次数由表达式来决定，当 n=101 就停止循环。

5.2.3 If…else…end 条件转移结构

语法：

```
if  条件式 1
       语句段 1
elseif  条件式 2
       语句段 2
       ...
else
       语句段 n+1
end
```

说明：当有多个条件时，条件式 1 为假再判断 elseif 的条件式 2，如果所有的条件式都不满足，则执行 else 的语句段 n+1，当条件式为真则执行相应的语句段；If…else…end 结构也可以是没有 elseif 和 else 的简单结构。

【例 5.6】用 If 结构执行二阶系统时域响应，根据阻尼系数 0<zeta<1 和 zeta=1 两种情况，得出不同的时域响应表达式。

```
function y=Ex0506(zeta)
% EX0506      使用 if 结构的二阶系统时域响应
x=0:0.1:20;
if (zeta>0)&(zeta<1)
    y=1-1/sqrt(1-zeta^2)*exp(-zeta*x).*sin(sqrt(1-zeta^2)*x+acos(zeta));
elseif zeta==1
    y=1-exp(-x).*(1+x);
end
plot(x,y)
```

5.2.4 switch…case 开关结构

语法：

```
switch  开关表达式
case  表达式 1
       语句段 1
case 表达式 2
       语句段 2
...
otherwise
       语句段 n
end
```

说明：

(1) 将开关表达式依次与 case 后面的表达式进行比较，如果表达式 1 不满足，则与下一个表达式 2 比较，如果都不满足则执行 otherwise 后面的语句段 n；一旦开关表达式与某个表达式相等，则执行其后面的语句段。

(2) 开关表达式只能是标量或字符串。

(3) case 后面的表达式可以是标量、字符串或元胞数组，如果是元胞数组则将开关表达式与元胞数组的所有元素进行比较，只要某个元素与开关表达式相等，就执行其后的语句段。

【例 5.7】用 switch…case 开关结构得出各月份的季节。

```
% EX0507      使用 switch 结构
for month=1:12;
    switch month
    case{3,4,5}
        season='spring'
    case{6,7,8}
        season='summer'
    case{9,10,11}
        season='autumn'
    otherwise
        season='winter'
    end
end
```

```
season =
winter
season =
winter
season =
spring
season =
spring
season =
spring
season =
summer
season =
summer
season =
summer
season =
autumn
season =
autumn
season =
autumn
season =
winter
```

程序分析：开关表达式为向量 1:12，case 后面的表达式为元胞数组，当元胞数组的某个元素与开关表达式相等，就执行其后的语句段。

5.2.5 try... catch... end 试探结构

语法：

```
try
    语句段 1
catch
    语句段 2
end
```

说明：首先试探性地执行语句段 1，如果在此段语句执行过程中出现错误，则将错误信息赋给保留的 lasterr 变量，并放弃这段语句，转而执行语句段 2 中的语句，当执行语句段 2 又出现错误，则终止该结构。

【例 5.8】用 try... catch... end 结构来进行矩阵相乘运算。

```
% EX0508     try 结构
n=4;
a=magic(n);
m=3;
b=eye(3);
try
    c=a*b
catch
    c=a(1:m,1:m)*b
end
lasterr
```

```
c =
    16     2     3
     5    11    10
     9     7     6
ans =
Error using ==> *
Inner matrix dimensions must agree.
```

程序分析：试探出矩阵的大小不匹配时，矩阵无法相乘，则再执行 catch 后面的语句段，将 a 的子矩阵取出与 b 矩阵相乘。可以通过这种结构灵活地实现矩阵的乘法运算。

5.2.6 流程控制语句

1. break 命令

break 命令可以使包含 break 的最内层的 for 或 while 语句强制终止，立即跳出该结构，执行 end 后面的命令，break 命令一般和 If 结构结合使用。

【例 5.9】将【例 5.5】增加条件用 If 与 break 命令结合，停止 while 循环。计算 1+3+5...+100 的值，当和大于 1000 时终止计算。

```
% EX0509    用 break 终止 while 循环
sum=0;
n=1;
while n<=100
    if sum<1000
        sum=sum+n;
        n=n+2;
    else
        break
    end
end
sum
n

sum =
        1024
n =
    65
```

程序分析：while…end 循环结构嵌套 If…else…end 分支结构，当 sum 为 1024 时跳出 while 循环结构，终止循环。

2. continue 命令

continue 命令用于结束本次 for 或 while 循环，只结束本次循环而继续进行下次循环。

【例 5.10】将 If 命令与 continue 命令结合，计算的 1～100 中所有素数的和，判断是否为素数是将 100 以内的每个数都被 2～$\sqrt{n}$ 整除，不能被整除的就是素数。

```
% EX0510    用 continue 终止 while 循环
sum=2;ss=0;
for n=3:100
    for m=2:fix(sqrt(n))
        if mod(n,m)==0
            ss=1;                  %能被整除就用 ss 为 1 表示
            break;                 %能被整除就跳出内循环
        else
            ss=0;                  %不能被整除就用 ss 为 0 表示
        end
    end
    if ss==1
            continue;              %能被整除就跳出本次外循
环
        end
    sum=sum+n;
 end
sum

sum =
```

```
        1060
```

程序分析：fix(sqrt(n))是将 $\sqrt{n}$ 取整；本程序为双重循环，两个 for 循环嵌套还嵌套一个 if 结构；当 mod(n,m)==0 时就用 break 跳出判断是否为素数的内循环，并继续用 continue 跳出求素数和的外循环而继续下次外循环。

3. return 命令

return 命令是终止当前命令的执行，并且立即返回到上一级调用函数或等待键盘输入命令，可以用来提前结束程序的运行。

注意：当程序进入死循环，则按 Ctrl+break 键来终止程序的运行。

4. pause 命令

pause 命令用来使程序运行暂停，等待用户按任意键继续。

语法：

```
pause        %暂停
pause(n)     %暂停 n 秒
```

5. keyboard 命令

keyboard 命令用来使程序暂停运行，等待键盘命令，执行完自己的工作后，输入 return 语句，程序就继续运行。

6. input 命令

input 命令用来提示用户应该从键盘输入数值、字符串和表达式，并接受该输入。

```
 a=input('input a number:')                %输入数值给 a
input a number:45
a =
    45
 b=input('input a number:','s')        %输入字符串给 b
input a number:45
b =
45
 input('input a number:')               %将输入值进行运算
input a number:2+3
ans =
     5
```

5.3 函数调用和参数传递

5.3.1 子函数和私有函数

1. 子函数

在一个 M 函数文件中，可以包含一个以上的函数，其中只有一个是主函数，其它则为子函数。

(1) 在一个 M 文件中，主函数必须出现在最上方，其后是子函数，子函数的次序无任何限制；

(2) 子函数不能被其它文件的函数调用，只能被同一文件中的函数(可以是主函数或子函数)调用；

(3) 同一文件的主函数和子函数变量的工作空间相互独立；

(4) 用 help 和 lookfor 命令不能提供子函数的帮助信息。

【例 5.11】将【例 5.2】画二阶系统时域曲线的函数作为子函数，编写画多条曲线的程序。

```
function Ex0511()
% EX0511       使用函数调用绘制二阶系统时域响应
z1=0.3;
Ex0502(z1);                  %调用 Ex0502
hold on
z1=0.5
Ex0502(z1)                   %调用 Ex0502
z1=0.707;
Ex0502(z1)                   %调用 Ex0502

function y=Ex0502(zeta)
%子函数，画二阶系统时域曲线
x=0:0.1:20;
y=1-1/sqrt(1-zeta^2)*exp(-zeta*x).*sin(sqrt(1-zeta^2)*x+acos(zeta))
plot(x,y)
```

程序分析：主函数是 Ex0511，子函数是 Ex0502，在主函数中三次调用子函数。程序保存为 Ex0511.m 文件。

2. 私有函数

私有函数是指存放在 private 子目录中的 M 函数文件，具有以下性质：

(1) 在 private 目录下的私有函数，只能被其父目录的 M 函数文件所调用，而不能被其它目录的函数调用，对其它目录的文件私有函数是不可见的，私有函数可以和其它目录下的函数重名；

(2) 私有函数父目录的 M 脚本文件也不可调用私有函数；

(3) 在函数调用搜索时，私有函数优先于其它 MATLAB 路径上的函数。

3. 调用函数的搜索顺序

在 MATLAB 中调用一个函数，搜索的顺序如下：

- 查找是否子函数；
- 查找是否私有函数；
- 从当前路径中搜索此函数；
- 从搜索路径中搜索此函数。

5.3.2 局部变量和全局变量

1. 局部变量

局部变量(Local Variables)是在函数体内部使用的变量，其影响范围只能在本函数内，只在函数执行期间存在。

2. 全局变量

全局变量(Global Variables)是可以在不同的函数工作空间和 MATALB 工作空间中共享使用的变量。

【例 5.12】修改【例 5.11】在主函数和子函数中使用全局变量。

```
function Ex0512()
% EX0512      使用全局变量绘制二阶系统时域响应
global X
X=0:0.1:20;
z1=0.3;
Ex0502(z1);
hold on
z1=0.5;
Ex0502(z1);
z1=0.707;
Ex0502(z1);

function Ex0502(zeta)
%子函数，画二阶系统时域曲线
global X
y=1-1/sqrt(1-zeta^2)*exp(-zeta*X).*sin(sqrt(1-zeta^2)*X+acos(zeta));
plot(X,y);
```

程序分析：X 变量为全局变量，在需要使用的主函数和子函数中都需要用 global 定义；同样如果在工作空间中定义 X 为全局变量后也可以使用：

```
 global X
who
Your variables are:
X
```

注意：由于全局变量在任何定义过的函数中都可以修改，因此不提倡使用全局变量；必须使用时应十分小心，建议把全局变量的定义放在函数体的开始，全局变量用大写字符命名。

5.3.3 函数的参数

1. 参数传递规则

【例 5.13】将【例 5.11】画二阶系统时域的函数修改，使用输入输出参数来实现参数传递，如图 5.3 所示。

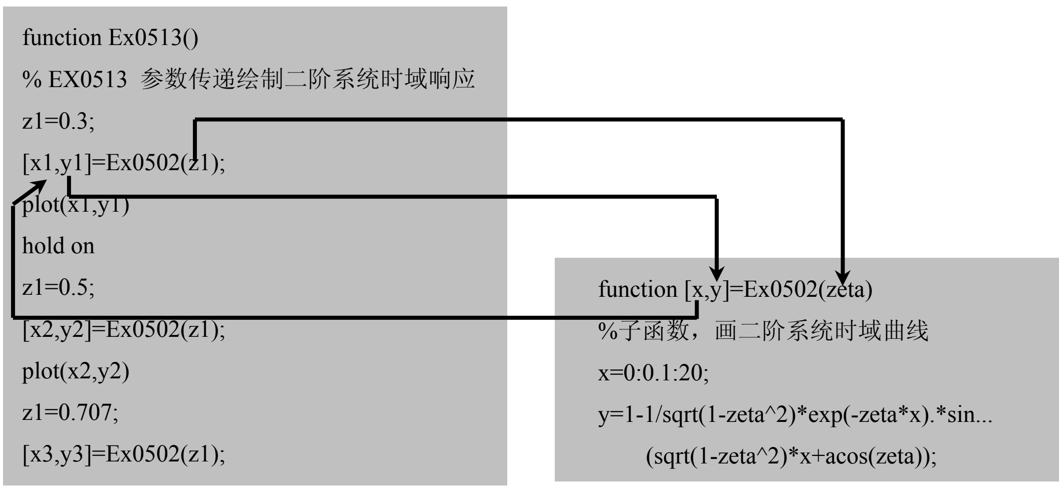


图 5.3 参数传递

程序分析：主函数 Ex0513 调用子函数 Ex0502，子函数中的 zeta 为输入参数，函数调用时将 z1 传递给子函数 zeta，子函数计算后将输出参数 x 和 y 传回给主函数的 x1、y1；主函数调用子函数三次，后面两次参数的传递也是同样。

2. 函数参数的个数

(1) nargin 和 nargout 变量

函数的输入输出参数的个数可以通过变量 nargin 和 nargout 获得，nargin 用于获得输入参数的个数，nargout 用于获得输出参数的个数。

语法：

```
nargin          %在函数体内获取实际输入变量的个数
nargout         %在函数体内获取实际输出变量的个数
nargin('fun')   %在函数体外获取定义的输入参数个数
nargout('fun')  %在函数体外获取定义的输出参数个数
```

【例 5.14】计算两个数的和，根据输入的参数个数不同使用不同的运算表达式。

```
function [sum,n]=Ex0514(x,y)
% EX0514    参数个数可变，计算 x 和 y 的和
if nargin==1
    sum=x+0;        %输入一个参数就计算与 0 的和
elseif nargin==0
    sum=0;          %无输入参数就输出 0
else
    sum=x+y;        %输入的是两个数则就计算和
end
```

在命令窗口调用 Ex0514 函数，分别使用 2 个、1 个和无输入参数结果如下：

```
 [y,n]=Ex0514(2,3)

y =
     5
n =
     2
 [y,n]=Ex0514(2)

y =
     2
n =
     1
 [y,n]=Ex0514

y =
     0
n =
     0
```

注意：如果输入的参数多于输入参数个数，则会出错。

```
 [y,n]=Ex0514(1,2,3)
??? Error using ==> ex0514
Too many input arguments.
```

也可以在工作空间查看函数体定义的输入参数个数：

```
 nargin('Ex0514')
ans =
     2
```

【例 5.14 续】添加以下程序，查看用 nargout 变量获取输出参数个数。

```
if nargout==0          %当输出参数个数为 0 时，运算结果为 0
    sum=0;
end
```

在命令窗口调用 Ex0514 函数，当输出参数格式不同时，结果如下：

```
 Ex0514(2,3)   %当输出参数个数为 0 时

ans =
     0
 y=Ex0514(2,3)    %当输出参数个数为 1 时

y =
     5
 [y,n,x]=Ex0514   %当输出参数个数为太多时

??? Error using ==> ex0514
Too many output arguments.
```

程序分析：当输出参数个数为 0 时，即使有两个输入参数，运算结果也为 0，结果送给 ans 变量；当输出的参数个数太多，也会出错。

(2) varargin 和 varargout 变量

varargin 和 varargout 可以获得输入输出变量的各元素内容。

【例 5.15】计算所有输入变量的和。

```
function [y,n]=Ex0515(varargin)
% EX0515     使用可变参数 varargin
if nargin==0                    %当没有输入变量时输出 0
    disp('No Input variables.')
    y=0;
elseif   nargin==1              %当一个输入变量时，输出该数
    y=varargin{1};
else
    n=nargin;
    y=0;
    for m=1:n
        y=varargin{m}+y;        %当有多个输入变量时，取输入变量循环相加
    end
end
n=nargin;
```

在 MATLAB 的命令窗口中输入不同个数的变量调用函数 Ex0515，结果如下：

```
 [y,n]=Ex0515(1,2,3,4)       %输入 4 个参数

y =
    10
```

```
n =
     4
 [y,n]=Ex0515(1)          %输入 1 个参数

y =
     1
n =
     1
 [y,n]=Ex0515         %无输入参数

No Input variables.
y =
     0
n =
     0
```

程序分析：n 为输入参数的个数；y 为求和运算的结果。

5.3.4 程序举例

【例 5.16】 根据阻尼系数绘制不同二阶系统的时域响应，当欠阻尼时 $y=1-\frac{1}{\sqrt{1-\varsigma^2}}e^{-\varsigma x}\sin(\sqrt{1-\varsigma^2}x+a\cos\varsigma)$，当临界阻尼时 $y=1-(1+x)e^{-x}$，当过阻尼时 $y=1-\frac{1}{2\sqrt{\varsigma^2-1}}\left(\frac{e^{-(\varsigma-\sqrt{\varsigma^2-1})x}}{\varsigma-\sqrt{1-\varsigma^2}}-\frac{e^{-(\varsigma+\sqrt{\varsigma^2-1})x}}{\varsigma+\sqrt{1-\varsigma^2}}\right)$。

M 文件的程序代码如下：

```
function y=Ex0516(z1)
% EX0516     主函数调用子函数，根据阻尼系数绘制二阶系统时域曲线
t=0:0.1:20;
if (z1>=0)&(z1<1)
    y=plotxy1(z1,t);
elseif z1==1
    y=plotxy2(z1,t);
else
    y=plotxy3(z1,t);
end

function y1=plotxy1(zeta,x)
%画欠阻尼二阶系统时域曲线
y1=1-1/sqrt(1-zeta^2)*exp(-zeta*x).*sin(sqrt(1-zeta^2)*x+acos(zeta));
plot(x,y1)

function y2=plotxy2(zeta,x)
%画临界阻尼二阶系统时域曲线
y2=1-exp(-x).*(1+x);
plot(x,y2)
```

```
function y3=plotxy3(zeta,x)
%画过阻尼二阶系统时域曲线
y3=1-1/(2*sqrt(zeta^2-1))*(exp(-((zeta-sqrt(zeta^2-1))*x))./(zeta-sqrt(zeta^2-1))...
     -exp(-((zeta+sqrt(zeta^2-1))*x))./(zeta+sqrt(zeta^2-1)));
plot(x,y3)
```

程序分析：主函数名为 Ex0516，三个子函数名分别为 plotxy1、plotxy2、plotxy3，文件保存为 Ex0516.m。当在命令窗口中输入以下命令：

```
y=Ex0516(0.3);
hold on
y=Ex0516(0.707);
y=Ex0516(1);
y=Ex0516(2);
```

则产生如图 5.4 所示时域响应曲线。

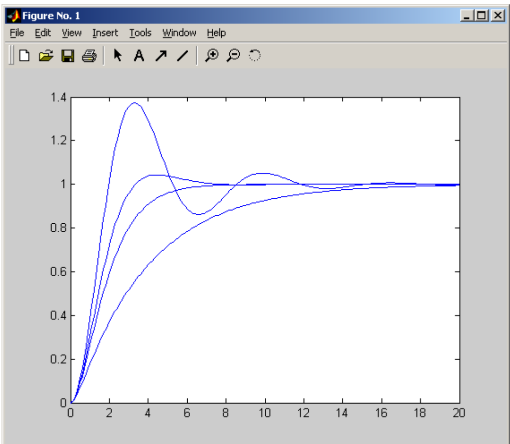


图 5.4 不同阻尼系数的时域响应曲线

【例 5.17】编写 M 函数文件，通过流程控制语句，建立如下的矩阵。

$$y=\begin{bmatrix}0 & 1 & 2 & 3 & \cdots & n\\ 0 & 0 & 1 & 2 & \cdots & n-1\\ 0 & 0 & 0 & 1 & \cdots & n-2\\ \vdots & \vdots & \vdots & \vdots & & \vdots\\ 0 & 0 & 0 & 0 & \cdots & 0\end{bmatrix}$$

```
function y=Ex0517(m)
% EX0517  用循环流程控制语句创建矩阵
y=0;
m=m-1;
for n=1:m
    y=[0,y];                    %创建全 0 行
end
for n=1:m
          a=[1:1:n];
     b=a;
           for k=m:-1:n
                b=[0,b];
     end
     y=[b;y];
```

```
        n=n+1;
    end
```

程序分析：将矩阵的行列数用输入参数 m 来确定，输出参数为矩阵 y。使用双重循环来创建矩阵，将文件保存为 Ex0517.m。

在命令窗口中调用 Ex0517 函数：

```
y=Ex0517(5)

y =
     0     1     2     3     4
     0     0     1     2     3
     0     0     0     1     2
     0     0     0     0     1
     0     0     0     0     0
```

5.4 M 文件性能的优化和加速

5.4.1 P 码文件

1. P 码文件的生成

P 码文件使用 pcode 命令生成，生成的 P 码文件与原 M 文件名相同，其扩展名为“.p”。

语法：

pcode Filename.m %在当前目录生成 Filename.p

pcode Filename.m -inplace %在 Filename.m 所在目录生成 Filename.p

```
pcode Ex0517.m
```

则在当前目录就生成了 P 码文件 Ex0517.p。

2. P 码文件的特点

(1) P 码文件的运行速度比原 M 文件速度快

(2) 存在同名的 M 文件和 P 码文件时则 P 码文件被调用

(3) P 码文件保密性好

用字处理软件打开 Ex0517.p 文件，看到的是乱码。

5.4.2 M 文件性能优化

1. 使用循环时提高速度的措施

循环语句及循环体是 MATLAB 编程的瓶颈问题，改进这种状况有三种方法：

(1) 尽量用向量的运算来代替循环操作。

(2) 在必须使用多重循环的情况下，如果两个循环执行的次数不同，则建议在循环的外环执行循环次数少的，内环执行循环次数多的，也可以显著提高速度。

(3) 应用 Mex 技术

如果耗时的循环不可避免，就应该考虑用其他语言，如 C 或 Fortran 语言，按照 Mex 技术要求的格式编写相应部分的程序，然后通过编译联接，形成在 MATLAB 可以直接调用的动态链接库(DLL)文件，这样就可以显著地加快运算速度(在 8.1.1 小节介绍)。

2. 大型矩阵的预先定维

给大型矩阵动态地定维是个很费时间的事。

【例 5.18】将【例 5.17】中的双重循环改为单循环，并先用 zeros 函数定维来提高运行速度，创建矩阵 $y=\begin{bmatrix}0 & 1 & 2 & 3 & \cdots & n\\ 0 & 0 & 1 & 2 & \cdots & n-1\\ 0 & 0 & 0 & 1 & \cdots & n-2\\ \vdots & \vdots & \vdots & \vdots & & \vdots\\ 0 & 0 & 0 & 0 & \cdots & 0\end{bmatrix}$。

```
function y=Ex0518(m)
% EX0518  先定维再创建矩阵
m=m-1;
y=zeros(m);
for n=1:m-1
            a=1:m-n;
      y(n,n+1:m)=a;
end
y
```

在命令窗口中分别调用 Ex0517 和 Ex0518 函数，比较其运行速度的不同：

```
y=Ex0517(200)
y=Ex0518(200)
```

3. 优先考虑内在函数

矩阵运算应该尽量采用 MATLAB 的内在函数，因为内在函数是由更底层的 C 语言构造的，其执行速度显然很快。

4. 采用高效的算法

在实际应用中，解决同样的数学问题经常有各种各样的算法。因此，应寻求更高效的算法。

5. 尽量使用 M 函数文件代替 M 脚本文件

由于 M 脚本文件每次运行时，都必须把程序装入内存，然后逐句解释执行，十分费时。

5.4.3 JIT 和加速器

1. JIT 和加速器的加速范围

JIT 和加速器的应用范围如下：

- 只对维数不超过 3 的“非稀疏”数组起作用；
- 只对“双精度”、“整数”、“字符串”和“逻辑”等四种数据类型起作用；
- 只对内部函数的调用起作用，对用户的 M 函数或 MEX 文件不起作用；
- 只对控制语句 for、while、if、elseif、switch 中标量运算的条件表达式起作用；
- 当一行程序中含有“不可加速的”命令或变量时，整行都不被加速；

- 当变量改变数据类型或维数，则该语句不被加速；
- 如果 i 和 j 不以虚数单位形式使用，则该语句不被加速；
- 在 Intel 系列 CPU 硬件上，Windows 和 Linux 系统加速能力最强。

2. 使用程序性能剖析窗口

(1) 打开程序性能剖析窗口

选择菜单“View”——“Profiler”；或使用在命令窗口输入“profile viewer”命令都可以打开程序性能剖析窗口，如图 5.5 所示。

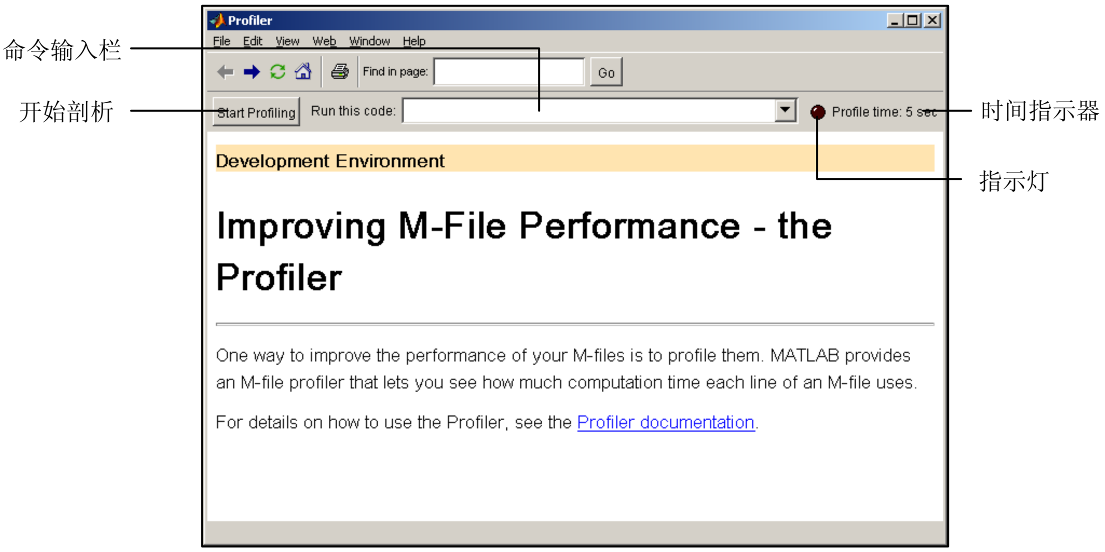


图 5.5 程序性能剖析窗口

(2) 对 MATLAB 的命令进行剖析

对 MATLAB 的命令进行剖析有两种方式：

- 在上图中的“命令输入栏”中输入需要剖析的命令，然后单击“Start Profiling”按钮，则指示灯变为绿色，“Start Profiling”按钮变灰色，时间指示器的计时在累加；当命令运行结束时，状态恢复，并出现剖析报告。
- 将“命令输入栏”清空，然后单击“Start Profiling”按钮开始剖析，则“Start Profiling”按钮变为“Stop Profiling”，指示灯变为绿色表示启动剖析，时间指示器的计时在累加；然后在 MATLAB 命令窗口中输入需要剖析的命令，进行剖析；当命令运行结束，单击“Stop Profiling”按钮停止剖析，则状态恢复，并出现剖析报告。这种方式的时间指示器显示的只是单击按钮“Stop Profiling”开始到单击“Stop Profiling”按钮的时间，并不表示命令的运行时间。

(3) 查看剖析报告

在程序性能分析窗口中以表格显示“剖析分析汇总表”(Profile Summary)，从上到下按占用时间的多少排列。

将第三章的例题进行剖析，【例 3.24】求微分方程组 $\frac{dx}{dt}=y, \frac{dy}{dt}=-x$ 的解，保存为“Ex0324.m”文件，内容如下：

```
%符号微分方程组求解
[x,y]=dsolve('Dx=y,Dy=-x')
[x,y]=dsolve('Dx=y,Dy=-x','t')
```

将该文件进行剖析，在“命令输入栏”中输入“Ex0324”，然后单击“Start Profiling”按钮开始剖析，则“剖析分析汇总表”如图 5.6 所示，单击“Ex0324”超链接则显示详细的列表：

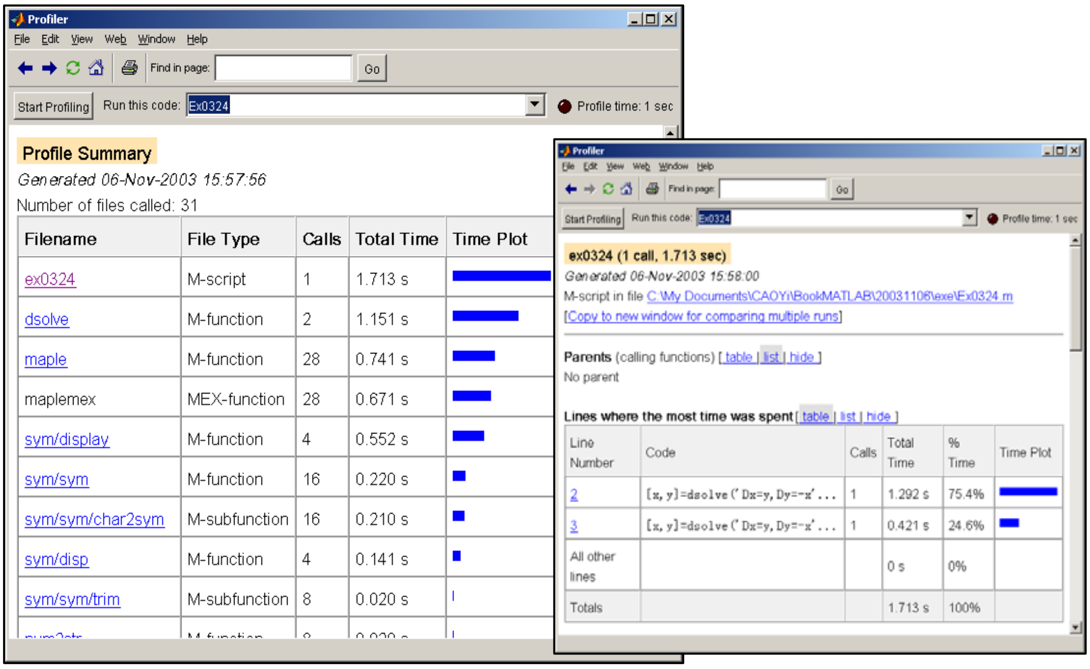


图 5.6 剖析报告

3. JIT 和加速器的开关函数

在 MATLAB6.5 版总是把 JIT 和加速器置于开启状态，可以使用命令来控制 JIT 和加速器的开启和关闭。

语法：

feature JIT on	%开启 JIT
feature JIT off	%关闭 JIT
feature accel on	%开启加速器
feature accel off	%关闭加速器

将【例 5.18】创建矩阵 $y=\begin{bmatrix}0 & 1 & 2 & 3 & \cdots & n\\ 0 & 0 & 1 & 2 & \cdots & n-1\\ 0 & 0 & 0 & 1 & \cdots & n-2\\ \vdots & \vdots & \vdots & \vdots & & \vdots\\ 0 & 0 & 0 & 0 & \cdots & 0\end{bmatrix}$ 的程序在开启或关闭 JIT 和加速器时，查看其运行速度。

在图 5.6 的“命令输入栏”中输入“y=Ex0518(200)”

当默认状态时 JIT 和加速器置于开启状态，在程序性能剖析窗口中查看运行时间为 0.43；当使用“feature JIT off”命令关闭 JIT 时，运行时间为 0.451；当将 JIT 和加速器都关闭时则运行时间为 0.481。

注意：JIT 和加速器对于不同的 M 文件作用不同，而不同的运行环境、不同软件、不同机器都会得出不同的运行时间，上面的运行时间只作为参考。

5.5 内联函数

1. 内联函数的创建

创建内联函数可以使用 inline 命令实现。

语法：

inline('string',arg1,arg2,…) %创建内联函数

说明：'string'必须是不带赋值号(“=”)的字符串；arg1 和 arg2 是函数的输入变量。

【例 5.19】创建内联函数实现 $f(x,z)=\sin(x)e^{-zx}$。

```
f=inline('sin(x)*exp(-z*x)','x','z')      %创建内联函数

f =
     Inline function:
     f(x,z) = sin(x)*exp(-z*x)
y=f(5,0.3)                             %调用函数 f

y =
   -0.2140
```

2. 查看内联函数

MATLAB 可以用 char、class 和 argnames 命令方便地查看内联函数的信息。

语法：

char(inline_fun) %查看内联函数的内容

class(inline_fun) %查看内联函数的类型

argnames(inline_fun) %查看内联函数的变量

【例 5.19 续】查看内联函数的信息。

```
char(f)

ans =
sin(x)*exp(-z*x)
class(f)

ans =
inline
argnames(f)

ans =
    'x'
    'z'
```

3. 使内联函数适用于数组运算

内联函数的输入变量不能是数组，但可以使用 vectorize 命令将内联函数适用于数组运算。

语法：

vectorize(inline_fun)　　%使内联函数适用于数组运算

【例 5.19 续】使内联函数适用于数组运算。

```
ff=vectorize(f)                %使内联函数 f 转换为适合数组运算

ff =
    Inline function:
    ff(x,z) = sin(x).*exp(-z.*x)
x=0:0.1:20;
y=ff(x,0.3);
```

4. 执行内联函数

内联函数还可以直接使用 feval 命令执行。

语法：

[y1，y2,…]=feval(inline_fun,arg1,arg2…)

【例 5.19 续】执行内联函数。

```
x=0:0.1:20;
z=0:0.05:10;
y=feval(ff,x,z)
```

5.6 利用函数句柄执行函数

5.6.1 函数句柄的创建

1. 函数句柄的创建

语法：

h_fun=@fun　　%创建函数句柄

h_fun=str2func(‘fun’)　　%创建函数句柄

h_array=str2func({‘fun1’,’fun2’,…})　　%创建函数句柄数组

说明：fun 是函数名，h_fun 是函数句柄，h_array 是函数句柄数组。

【例 5.20】创建 MATLAB 内部函数的句柄。

```
h_sin=@sin;                         %创建函数句柄
h_cos=str2func('cos');                     %创建函数句柄数组
h_array=str2func({'sin','cos','tan'})

h_array =
    @sin     @cos     @tan
```

2. 使用函数句柄的优点

(1) 在更大范围调用函数

函数句柄包含了函数文件的路径和函数类型，即函数是否为内部函数、M 或 P 文件、子函数、私有函数等，因此无论函数所在的文件是否在搜索路径上，是否是当前路径，是否是子函数或私有函数，只要函数句柄存在，函数就能执行。

(2) 提高函数调用的速度

不使用函数句柄时，对函数的每次调用都要为该函数进行全面的路径搜索，直接影响了速度。

(3) 使函数调用象使用变量一样方便、简单。

(4) 可迅速获得同名重载函数的位置、类型信息。

5.6.2 用 feval 命令执行函数

函数也可以使用 feval 命令直接执行，feval 命令可以使用函数句柄或函数名。

语法：

[y1,y2,…]=feval(h_fun,arg1,arg2…)

[y1,y2,…]=feval('funname',arg1,arg2…)

说明：h_fun 是函数句柄，'funname'是函数名，arg1、arg2…是输入参数，y1、y2…是输出参数。

【例 5.21】将【例 5.17】编写的绘制二阶系统时域响应曲线中的调用各子函数改为利用函数句柄实现。

```
function y=Ex0521(z1)
% EX0521    利用函数句柄执行函数，二阶系统时域响应
t=0:0.1:20;
h_plotxy1=str2func('plotxy1')   %创建函数句柄
h_plotxy2=str2func('plotxy2')   %创建函数句柄
h_plotxy3=str2func('plotxy3')   %创建函数句柄
if (z1>=0)&(z1<1)
    y=feval(h_plotxy1,z1,t);   %执行函数
elseif z1==1
    y=feval(h_plotxy2,z1,t);   %执行函数
else
    y=feval(h_plotxy3,z1,t);   %执行函数
end

function y1=plotxy1(zeta,x)
%画欠阻尼二阶系统时域曲线
y1=1-1/sqrt(1-zeta^2)*exp(-zeta*x).*sin(sqrt(1-zeta^2)*x+acos(zeta));
plot(x,y1)

function y2=plotxy2(zeta,x)
%画临界阻尼二阶系统时域曲线
y2=1-exp(-x).*(1+x);
plot(x,y2)

function y3=plotxy3(zeta,x)
%画过阻尼二阶系统时域曲线
y3=1-1/(2*sqrt(zeta^2-1))*(exp(-((zeta-sqrt(zeta^2-1))*x))./(zeta-sqrt(zeta^2-1))...
    -exp(-((zeta+sqrt(zeta^2-1))*x))./(zeta+sqrt(zeta^2-1)));
plot(x,y3)
```

程序分析：在主函数中使用函数句柄执行各子函数，运行的结果与【例 5.17】相同。

在 MATLAB 的命令窗口调用该 Ex0521 函数有三种格式：

(1) 用 feval 命令利用函数句柄执行

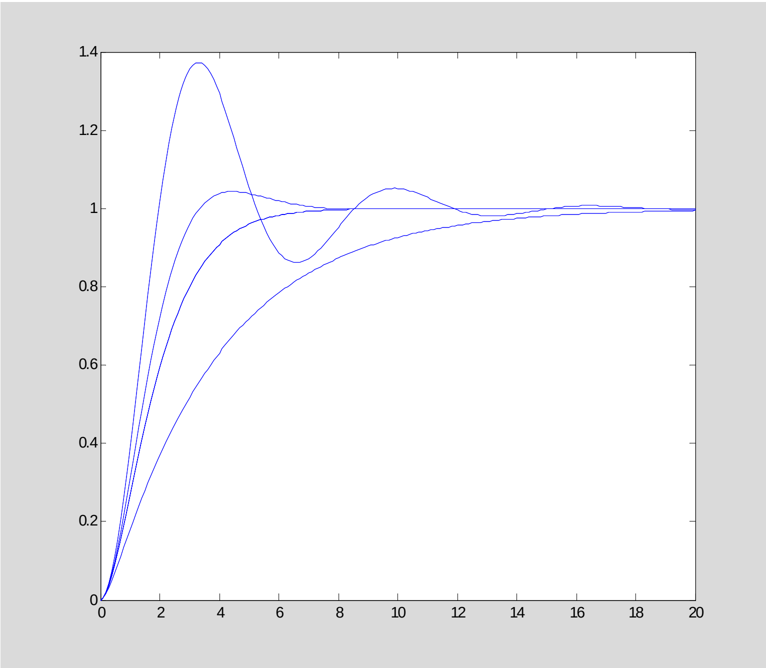
```
h_Ex0521=str2func('Ex0521')
```

```
h_Ex0521 =
    @Ex0521
 y=feval(h_Ex0521,1);

h_plotxy1 =
    @plotxy1
h_plotxy2 =
    @plotxy2
h_plotxy3 =
    @plotxy3
```

(2) 用 feval 命令利用函数名执行

```
 y=feval('Ex0521',1);
```



(3) 直接调用函数

```
 y=Ex0521(1);
```

5.7 利用泛函命令进行数值分析

在 MATLAB 中，所有以函数为输入变量的命令，都称为泛函命令。

常见语法：

[输出变量列表]=函数名(h_fun,输入变量列表)

[输出变量列表]=函数名('funname',输入变量列表)

说明：h_fun 是要被执行的 M 函数文件的句柄，或者是内联函数和字符串；'funname' 是 M 函数文件名。

5.7.1 求极小值

1. fminbnd 函数

fminbnd 函数用来计算单变量非线性函数的最小值。

语法：

[x,y]=fminbnd(h_fun,x1,x2,options)

[x,y]=fminbnd('funname',x1,x2,options)

说明：h_fun 是函数句柄，'funname'是函数名，必须是单值非线性函数；options 是用来控制算法的参数向量，默认值为 0 可省略；x 是 fun 函数在区间 x1<x<x2 上的局部最小值的发生点；y 是对应的最小值。

【例 5.22】用 fminbnd 求解 humps 函数的极小值。

```
[x,y]=fminbnd(@humps,0.5,0.8)                    %求在 0.5—0.8 之间极小值

x =
    0.6370
y =
   11.2528
```

程序分析：humps 函数是 MATLAB 提供的 M 文件，保存为 humps.m 文件，@humps 表示 humps 函数的句柄，humps 的曲线如图 5.7 所示，最小值为图中的圆点(0.6370，11.2528)，误差小于 10^{-4}。

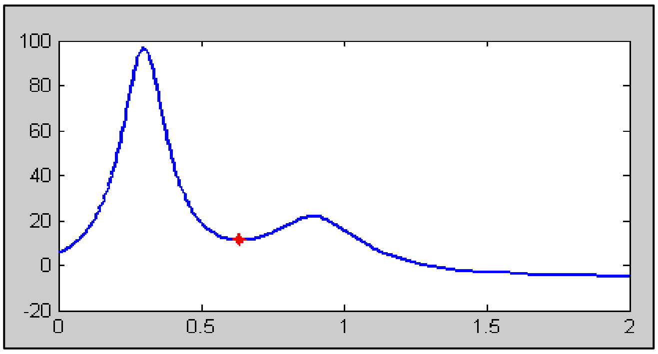


图 5.7 求 humps 函数最小值

2. fminsearch 函数

fminsearch 函数是求多变量无束缚非线性最小值。

语法：

x=fminsearch(h_fun,x0)

x=fminsearch('funname',x0)

说明：x0 是最小值点的初始猜测值。

【例 5.23】求著名的 Banana 测试函数 $f(x,y)=100(y-x^2)^2+(1-x)^2$ 的最小值，它的理论最小值是 x=1，y=1。该测试函数有一片浅谷，很多算法都难以逾越。

```
fn=inline('100*(x(2)-x(1)^2)^2+(1-x(1))^2' ,'x')     %用 inline 产生内联函数,x 和 y 用二元数组表示
```

```
fn =
     Inline function:
     fn(x) = 100*(x(2)-x(1)^2)^2+(1-x(1))^2
```

```
y=fminsearch(fn,[0.5,-1])                  %从(0.5,-1)为初始值开始搜索求最小值

y =
    1.0000    1.0000
```

5.7.2 求过零点

fzero 函数可以寻找一维函数的零点，即求 f(x)=0 的根。

语法：

x=fzero(h_fun,x0,tol,trace)

x=fzero('funname',x0,tol,trace)

说明：h_fun 是待求零点的函数句柄；x0 有两个作用：预定待搜索零点的大致位置和搜索起始点；tol 用来控制结果的相对精度，默认值为 eps；trace 指定迭代信息是否在运算中显示，默认为 0，表示不显示迭代信息。tol 和 trace 都可以省略。

【例 5.24】求解 humps 函数的过零点，humps 函数如图 5.8 所示，过零点用圆点表示。

```
xzero=fzero(@humps,1)              %求在 1 附近的零点

xzero =
    1.2995
xzero=fzero(@humps,[0.5,1.5])          %求在 0.5—1.5 范围的零点

xzero =
    1.2995
xzero=fzero(@humps,[0.5,1])          %求在 0.5—1 范围的零点

??? Error using ==> fzero
The function values at the interval endpoints must differ in sign.
```

程序分析：当在 0.5～1 的范围找不到零点，提示出错信息。

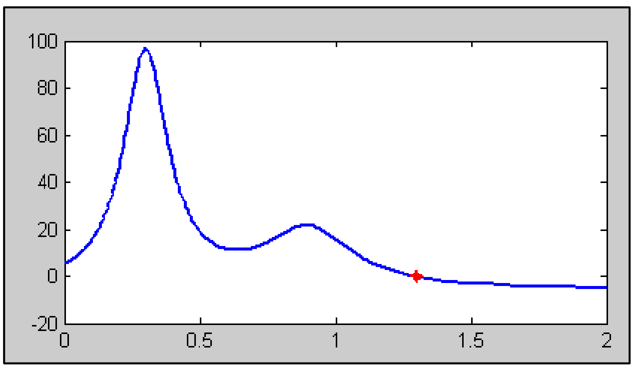


图 5.8 求 humps 函数的过零点

5.7.3 数值积分

函数 quad 和 quad8 是基于数学上的正方形概念来计算函数的面积，quad8 比 quad 更精确速度更快。

语法：

s=quad(h_fun,x1,x2,tol,trace,p1,p2,…)

s=quad('funname',x1,x2,tol,trace,p1,p2,…)

s=quad8(h_fun,x1,x2,tol,trace,p1,p2,…)

s=quad8('funname',x1,x2,tol,trace,p1,p2,…)

说明：x1 和 x2 分别是积分的上、下限；tol 用来控制积分精度，省略时默认为 0.001；trace 取 1 用图形展现积分过程，取 0 则无图形，省略时默认不画图；p1,p2…是向函数传递的参数，可省略。

【例 5.25】计算 y=humps(x)曲线下面的面积。

```
x=0:0.01:1;
y=humps(x);
area=trapz(x,y)                 %用梯形计算积分

area =
   29.8571
area1=quad(@humps,0,1)          %用 quad 计算积分

area1 =
   29.8583
area2=quad8(@humps,0,1)      %用 quad8 计算积分

Warning: QUAD8 is obsolete.  QUADL is its recommended replacement.
(Type "warning off MATLAB:quad8:ObsoleteFunction" to suppress this
warning.)
> In D:\MATLAB\toolbox\matlab\funfun\quad8.m at line 35
area2 =
   29.8583
```

程序分析：用 trapz 函数梯形近似可能低估了实际面积，如果当梯形的宽度变窄时，就能够得到更精确的结果。quad 和 quad8 函数返回非常相近的估计面积。

5.7.4 微分方程的数值解

MATLAB 提供 ode23、ode45 和 ode113 等多个函数求解微分方程的数值解：

- 低维方法解一阶常微分方程组

语法：

[t,y]=ode23(h_fun,tspan,y0,options,p1,p2…)

[t,y]=ode23('funname',tspan,y0,options,p1,p2…)

- 高维方法解一阶常微分方程组

语法：

[t,y]=ode45(h_fun,tspan,y0,options,p1,p2…)

[t,y]=ode45('funname',tspan,y0,options,p1,p2…)

- 变维方法解一阶常微分方程组

语法：

[t,y]=ode113(h_fun,tspan,y0,options,p1,p2…)

[t,y]=ode113('funname',tspan,y0,options,p1,p2…)

说明：h_fun 是函数句柄，函数以 dx 为输出，以 t,y 为输入量；tspan=[起始值 终止值]，表示积分的起始值和终止值；y0 是初始状态列向量；options 可以定义函数运行时的参数，可省略；p1,p2…是函数的输入参数，可省略。

【例 5.26】解经典的范德波尔(Van der Pol)微分方程：

$$\frac{d^2x}{dt^2}-\mu(1-x^2)\frac{dx}{dt}+x=0$$

(1) 必须把高阶微分方程式变换成一阶微分方程组。

令 y1=x，y2=dx/dt，则将二阶微分方程变为一阶微分方程组：

$$\begin{bmatrix}\frac{dy_1}{dt}\\ \frac{dy_2}{dt}\end{bmatrix}=\begin{bmatrix}y_2\\ \mu(1-y_1^2)y_2-y_1\end{bmatrix}$$

(2) 编写一个函数 vdpol.m 文件，设定 μ =2，该函数返回上述导数值。输出结果由一个列向量 yprime 给出。y1 和 y2 合并写成列向量 y。

函数 M 文件 vdpol.m：

```
%范德波尔方程
function yprime=vdpol(t,y)
yprime=[y(2);2*(1-y(1)^2)*y(2)-y(1)]
```

(3) 给定当前时间及 y1 和 y2 的初始值，解微分方程：

```
 tspan=[0,30];                    %起始值 0 和终止值 30
 y0=[1;0];                        %初始值
 [t,y]=ode45(@vdpol,tspan,y0);    %解微分方程
 y1=y(:,1);
 y2=y(:,2);
 figure(1)
plot(t,y1,':b',t,y2,'-r')         %画微分方程解
figure(2)
plot(y1,y2)                       %画相平面图
```

则微分方程 y1 和 y2 在时间域的曲线如图 5.9 所示。

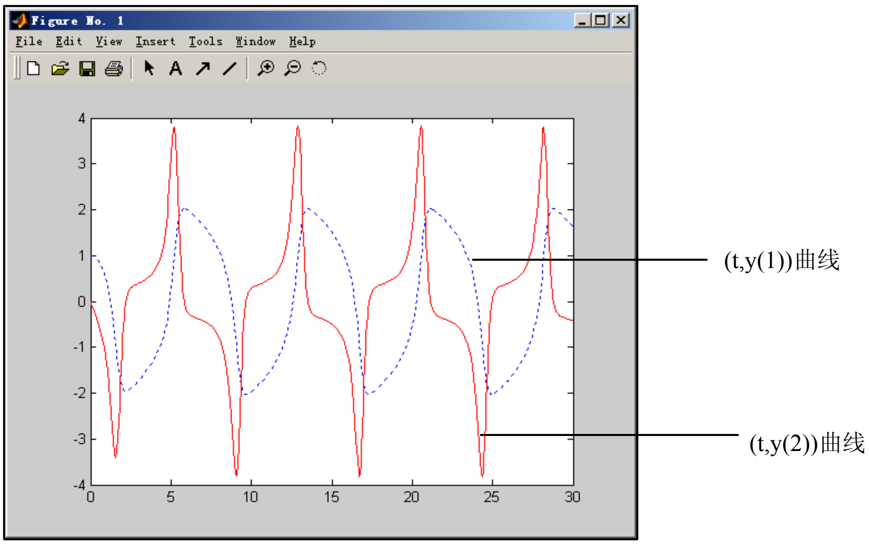


图 5.9 微分方程解

将 y(1)为横坐标，y(2)为纵坐标则为相平面图，如图 5.10 所示。

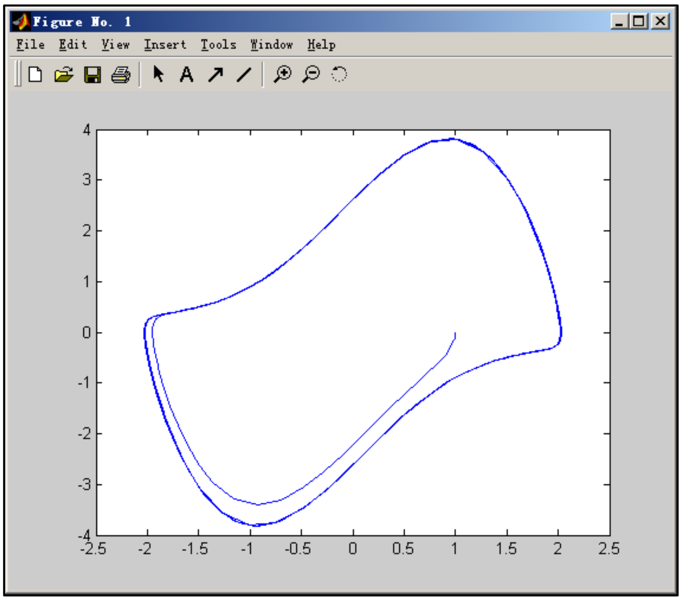


图 5.10 相平面图

第 6 章 线性控制系统分析与设计

MATLAB 的控制系统工具箱(Control System Toolbox)可以提供对线性系统分析、设计和建模的各种算法。

6.1 线性系统的描述

6.1.1 状态空间描述法

状态空间描述法是使用状态方程模型来描述控制系统，MATLAB 中状态方程模型的建立使用 ss 和 dss 命令。

语法：

G=ss(a,b,c,d) %由 a、b、c、d 参数获得状态方程模型

G=dss(a,b,c,d,e) %由 a、b、c、d、e 参数获得状态方程模型

【**例 6.1**】写出二阶系统 $\frac{d^2y(t)}{dt^2}+2\zeta\omega_n\frac{dy(t)}{dt}+\omega_n{}^2y(t)=\omega_n{}^2u(t)$，当 ζ=0.707，ω_n=1 时的状态方程。

```
zeta=0.707;wn=1;
A=[0 1;-wn^2 -2*zeta*wn];
B=[0;wn^2];
C=[1 0];
```

```
D=0;
G=ss(A,B,C,D)        %建立状态方程模型
```

```
a =
          x1      x2
   x1      0       1
   x2     -1  -1.414


b =
       u1
   x1   0
   x2   1


c =
       x1  x2
   y1   1   0


d =
       u1
   y1   0

Continuous-time model.
```

6.1.2 传递函数描述法

MATLAB 中使用 tf 命令来建立传递函数。

语法：

G=tf(num,den)　　　%由传递函数分子分母得出

说明：num 为分子向量，num=$[b_1,b_2,\ldots,b_m,b_{m+1}]$；den 为分母向量，den=$[a_1,a_2,\ldots,a_{n-1},a_n]$。

【**例 6.1 续**】将二阶系统描述为传递函数的形式。

```
num=1;
den=[1 1.414 1];
G=tf(num,den)        %得出传递函数
```

```
Transfer function:
       1
-----------------
s^2 + 1.414 s + 1
```

6.1.3 零极点描述法

MATLAB 中使用 zpk 命令可以来实现由零极点得到传递函数模型。

语法：

G=zpk(z,p,k)　　　　%由零点、极点和增益获得

说明：z 为零点列向量；p 为极点列向量；k 为增益。

【例 6.1 续】得出二阶系统的零极点，并得出传递函数。

```
 z=roots(num)

z =
   Empty matrix: 0-by-1
 p=roots(den)

p =
  -0.7070 + 0.7072i
  -0.7070 - 0.7072i
 zpk(z,p,1)


Zero/pole/gain:
        1
-------------------
(s^2  + 1.414s + 1)
```

程序分析：roots 函数可以得出多项式的根，零极点形式是以实数形式表示的。

部分分式法是将传递函数表示成部分分式或留数形式：

$$G(s)=\frac{r_1}{s-p_1}+\frac{r_2}{s-p_2}+\cdots+\frac{r_n}{s-p_n}+k(s)$$

【例 6.1 续】将传递函数转换成部分分式法，得出各系数。

```
 [r,p,k]=residue(num,den)

r =
        0 - 0.7070i
        0 + 0.7070i
p =
  -0.7070 + 0.7072i
  -0.7070 - 0.7072i
k =
     []
```

6.1.4 离散系统的数学描述

1.　状态空间描述法

状态空间描述离散系统也可使用 ss 和 dss 命令。

语法：

G=ss(a,b,c,d,Ts) %由 a、b、c、d 参数获得状态方程模型

G=dss(a,b,c,d,e,Ts) %由 a、b、c、d、e 参数获得状态方程模型

说明：Ts 为采样周期，为标量，当采样周期未指明可以用-1 表示。

【例 6.2】用状态空间法建立离散系统。

```
a=[-1.5 -0.5;1 0];
b=[1;0];
c=[0 0.5];
d=0;
G=ss(a,b,c,d,0.1)        %采样周期为 0.1s
```

```
a =
          x1    x2
   x1   -1.5  -0.5
   x2      1     0


b =
        u1
   x1    1
   x2    0


c =
         x1   x2
   y1     0  0.5


d =
        u1
   y1    0

Sampling time: 0.1
Discrete-time model.
```

2. 脉冲传递函数描述法

脉冲传递函数也可以用 tf 命令实现。

语法：

G=tf(num,den,Ts) %由分子分母得出脉冲传递函数

说明：Ts 为采样周期，为标量，当采样周期未指明可以用-1 表示，自变量用'z'表示。

【例 6.2 续】创建离散系统脉冲传递函数 $G(z)=\dfrac{0.5z}{z^2-1.5z+0.5}=\dfrac{0.5z^{-1}}{1-1.5z^{-1}+0.5z^{-2}}$。

```
num1=[0.5 0];
```

```
den=[1 -1.5 0.5];
G1=tf(num1,den,-1)
```

```
Transfer function:
     0.5 z
-----------------
z^2 - 1.5 z + 0.5

Sampling time: unspecified
```

MATLAB 中还可以用 filt 命令产生脉冲传递函数。

语法：

G=filt(num,den,Ts)　　　　%由分子分母得出脉冲传递函数

说明：Ts 为采样周期，当采样周期未指明 Ts 可以省略，也可以用-1 表示，自变量用'z^{-1}'表示。

【例 6.2 续】使用 filt 命令产生脉冲传递函数。

```
num2=[0 0.5];
G2=filt(num2,den)
```

```
Transfer function:
      0.5 z^-1
-----------------------
1 - 1.5 z^-1 + 0.5 z^-2

Sampling time: unspecified
```

程序说明：用 filt 命令生成的脉冲传递函数的自变量不是 z 而是 z^{-1}，因此分子应改为“[0 0.5]”。

3. 零极点增益描述法

离散系统的零极点增益用 zpk 命令实现。

语法：

G=zpk(z,p,k,Ts)　　　　%由零极点得出脉冲传递函数

【例 6.2 续】使用 zpk 命令产生零极点增益传递函数。

```
G3=zpk([0],[0.5 1],0.5,-1)
```

```
Zero/pole/gain:
   0.5 z
-------------
(z-0.5) (z-1)

Sampling time: unspecified
```

6.2 线性系统模型之间的转换

6.2.1 连续系统模型之间的转换

在 MATLAB5.3 版及以前的控制系统工具箱中有各种不同模型转换的函数，如下表 6.1 所示为线性系统模型转换的函数。

表 6.1 线性系统模型转换函数表

函数	调用格式	功能
tf2ss	[a,b,c,d]=tf2ss(num,den)	传递函数转换为状态空间
tf2zp	[z,p,k]=tf2zp(num,den)	传递函数转换为零极点描述
ss2tf	[num,den]=ss2tf(a,b,c,d,iu)	状态空间转换为传递函数
ss2zp	[z,p,k]=ss2zp(a,b,c,d,iu)	状态空间转换为零极点描述
zp2ss	[a,b,c,d]=zp2ss(z,p,k)	零极点描述转换为状态空间
zp2tf	[num,den]=zp2tf(z,p,k)	零极点描述转换为传递函数

1. 系统模型的转换

(1) 状态空间模型的获得

由命令 ss 和 dss 实现将传递函数和零极点增益转换为状态空间模型。

语法：

G=ss(传递函数)　　　　%由传递函数转换获得

G=ss(零极点模型)　　　%由零极点模型转换获得

【例 6.3】将单输入双输出的系统传递函数 $G_1(s)=\frac{\begin{bmatrix}3s+2\\ s^2+2s+5\end{bmatrix}}{3s^3+5s^2+2s+1}$ 转换为状态空间描述。

```
 num=[0 3 2;
1 2 3];
 den=[3 5 2 1];
 G11=tf(num(1,:),den)


Transfer function:
       3 s + 2
-----------------------
3 s^3 + 5 s^2 + 2 s + 1

  G12=tf(num(2,:),den)


Transfer function:
    s^2 + 2 s + 3
-----------------------
```

```
3 s^3 + 5 s^2 + 2 s + 1

 G=ss([G11;G12])


a =
            x1        x2        x3
   x1    -1.667   -0.3333  -0.08333
   x2         2         0         0
   x3         0         2         0


b =
       u1
   x1   1
   x2   0
   x3   0


c =
          x1      x2      x3
   y1      0     0.5  0.1667
   y2  0.3333  0.3333    0.25


d =
       u1
   y1   0
   y2   0

Continuous-time model.
```

(2) 传递函数的获得

由 tf 命令实现将系统的状态空间法和零极点增益模型转换为传递函数。

语法：

G=tf(状态方程模型) %由状态空间转换

G=tf(零极点模型) %由零极点模型转换

【例 6.3 续】由状态空间描述转换为传递函数。

```
 G1=tf(G)


Transfer function from input to output...
                s + 0.6667
 #1:  -----------------------------------
      s^3 + 1.667 s^2 + 0.6667 s + 0.3333
```

```
         0.3333 s^2 + 0.6667 s + 1
 #2:  -----------------------------------
     s^3 + 1.667 s^2 + 0.6667 s + 0.3333
```

(3) 零极点模型的获得

由 zpk 命令实现将状态空间法、传递函数转换为零极点模型。

语法：

G=zpk(状态方程模型)　　　　%由状态方程模型转换

G=zpk(传递函数)　　　　%由传递函数转换

【例 6.3 续】由传递函数和状态方程模型转换零极点模型。

```
 G2=zpk(G)                %由状态方程模型转换


Zero/pole/gain from input to output...
               (s+0.6667)
 #1:  -----------------------------------
     (s+1.356) (s^2  + 0.3103s + 0.2458)

          0.33333 (s^2  + 2s + 3)
 #2:  -----------------------------------
     (s+1.356) (s^2  + 0.3103s + 0.2458)

  G2=zpk(G1);             %由传递函数转换
```

2. 模型参数的获取

语法：

[a,b,c,d]=ssdata(G)　　　　%获取状态空间参数

[a,b,c,d,e]=dssdata(G)　　　　%获取状态空间参数

[num,den]=tfdata(G)　　　　%获取传递函数参数

[z,p,k]=zpkdata(G)　　　　%获取零极点参数

【例 6.3 续】获取各模型的参数。

```
 [a,b,c,d]=ssdata(G1)              %获取状态方程参数

a =
   -1.6667   -0.3333   -0.0833
    2.0000         0         0
         0    2.0000         0
b =
     1
     0
     0
c =
         0    0.5000    0.1667
```

```
    0.3333    0.3333    0.2500
d =
     0
     0
 [num,den]=tfdata(G2)           %获取传递函数参数

num =
    [1x4 double]
    [1x4 double]
den =
    [1x4 double]
    [1x4 double]
 [z,p,k]=zpkdata(G)          %获取零极点参数

z =
    [   -0.6667]
    [2x1 double]
p =
    [3x1 double]
    [3x1 double]
k =
    1.0000
    0.3333
```

3.　模型类型的检验

【**例 6.3 续**】检验模型的类型。

```
 class(G)          %得出系统模型类型

ans =
ss
 isa(G,'tf')       %检验系统模型类型

ans =
     0
```

6.2.2 连续系统与离散系统之间的转换

表 6.2　模型类型检验函数表

函数	调用格式	功能
class	class(G)	得出系统模型的类型
isa	isa(G, '类型名')	判断 G 是否对应的类型名，是则为 1(True)
isct	isct(G)	判断 G 是否连续系统，是则为 1(True)
isdt	isdt(G)	判断 G 是否离散系统，是则为 1(True)

issiso	issiso(G)	判断 G 是否 SISO 系统，是则为 1(True)

1.　c2d 命令

c2d 命令用于将连续系统转换为离散系统。

语法：

Gd=c2d(G,Ts,method)　　　　%以采样周期 Ts 和 method 方法转换为离散系统

说明：G 为连续系统模型；Gd 为离散系统模型；Ts 为采样周期；method 为转换方法，可省略，包括五种：zoh(默认零阶保持器)、foh(一阶保持器)、tustin(双线性变换法)、prewarp(频率预修正双线性变换法)、mached(根匹配法)。

【例 6.4】将二阶连续系统转换为离散系统。

```
a=[0 1;-1 -1.414];
b=[0;1];
c=[1 0];
d=0;
G=ss(a,b,c,d);
Gd=c2d(G,0.1)
```

```
a =
             x1       x2
   x1    0.9952   0.0931
   x2   -0.0931   0.8636


b =
              u1
   x1   0.004768
   x2     0.0931


c =
        x1   x2
   y1    1    0


d =
        u1
   y1    0

Sampling time: 0.1
Discrete-time model.
```

2.　d2c 命令

d2c 命令是 c2d 的逆运算，用于将离散系统转换为连续系统。

语法：

G=d2c(Gd,method)　　　　%转换为连续系统

说明：method 为转换方法可省略，与 c2d 相似，少了 foh(一阶保持器)方法。

【**例 6.4 续**】将二阶离散系统转换为连续系统。

```
G=d2c(Gd)


a =
                 x1          x2
   x1  5.551e-016           1
   x2          -1      -1.414


b =
                  u1
   x1  -2.776e-016
   x2            1


c =
        x1  x2
   y1    1   0


d =
        u1
   y1    0

Continuous-time model.
```

3. d2d 命令

d2d 命令是将离散系统改变采样频率。

语法：

Gd2=d2d(Gd1,Ts2)　　　　%转换离散系统的采样频率为 Ts2

说明：其实际的转换过程是先把 Gd1 按零阶保持器转换为原连续系统，然后再用 Ts2 和零阶保持器转换为 Gd2。

【**例 6.4 续**】将二阶离散系统改变采样频率。

```
Gd2=d2d(Gd,0.3)


a =
              x1       x2
   x1     0.961   0.2408
   x2   -0.2408   0.6205
```

```
b =
            u1
   x1  0.03897
   x2   0.2408


c =
       x1  x2
   y1   1   0


d =
       u1
   y1   0

Sampling time: 0.3
Discrete-time model.
```

6.2.3 模型对象的属性

1. 模型对象的属性

ss、tf 和 zpk 三种对象除了具有线性时不变系统共有的属性以外，还具有其各自的属性，共有属性如表 6.3 所示，其各自的属性如表 6.4 所示。

表 6.3 对象共有属性表

属性名	属性值的数据类型	意义
Ts	标量	采样周期，为 0 表示连续系统，为-1 表示采样周期未定
Td	数组	输入延时，仅对连续系统有效，省略表示无延时
InputName	字符串数组	输入变量名
OutputName	字符串数组	输出变量名
Notes	字符串	描述模型的文本说明
Userdata	任意数据类型	用户需要的其它数据

表 6.4 三种子对象特有属性表

对象名	属性名	属性值的数据类型	意义
tf	den	行数组组成的单元阵列	传递函数分母系数
	num	行数组组成的单元阵列	传递函数分子系数
	variable	s,p,z,q,z^{-1} 之一	传递函数变量
ss	a	矩阵	系数
	b	矩阵	系数
	c	矩阵	系数
	d	矩阵	系数
	e	矩阵	系数

	StateName	字符串向量	用于定义每个状态变量的名称
zpk	z	矩阵	零点
	p	矩阵	极点
	k	矩阵	增益
	variable	s,p,z,q,z^{-1}之一	零极点增益模型变量

在表 6.3 和表 6.4 中的三种子对象的属性，在前面都已使用过，MATLAB 提供了 get 和 set 命令来对属性进行获取和修改。

2. get 命令和 set 命令

(1) get 命令可以获取模型对象的所有属性

语法：

get(G) %获取对象的所有属性值

get(G,’PropertyName’,…) %获取对象的某些属性值

说明：G 为模型对象名；’PropertyName’为属性名。

(2) set 命令用于修改对象属性名

语法：

set(G,’PropertyName’,PropertyValue,…) %修改对象的某些属性值

【**例 6.5**】已知二阶系统的传递函数 $G(s)=\dfrac{1}{s^2+1.414s+1}$，获取其传递函数模型的属性，并将传递函数修改为 $\dfrac{1}{z^2+2z+1}$。

```
num=1;
den=[1 1.414 1];
G=tf(num,den);
get(G)                              %获取所有属性

         num: {[0 0 1]}
         den: {[1 1.41 1]}
    Variable: 's'
          Ts: 0
     ioDelay: 0
  InputDelay: 0
 OutputDelay: 0
   InputName: {''}
  OutputName: {''}
  InputGroup: {0x2 cell}
 OutputGroup: {0x2 cell}
       Notes: {}
    UserData: []
set(G,'den',[1 2 1],'Variable','z')          %设置属性
G


Transfer function:
```

```
     1
-------------
z^2 + 2 z + 1

Sampling time: unspecified
```

3. 直接获取和修改属性

【例 6.5 续】将上面的传递函数模型对象的分母修改为原来的值。

```
G.den=[1 1.414 1];
G
```

```
Transfer function:
       1
-----------------
z^2 + 1.414 z + 1

Sampling time: unspecified
```

6.3 结构框图的模型表示

1. 串联结构

SISO 的串联结构是两个模块串联在一起，如图 6.1 所示。

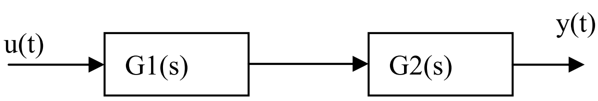


图 6.1 串联结构

实现串联结构传递函数的命令：

G=G1*G2

G=series(G1,G2)

2. 并联结构

SISO 的并联结构是两个模块并联在一起，如图 6.2 所示。

实现并联结构传递函数的命令：

G=G1+G2

G=parallel(G1,G2)

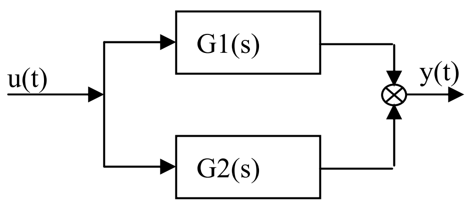


图 6.2 并联结构

3. 反馈结构

反馈结构是前向通道和反馈通道模块构成正反馈和负反馈，如图 6.3 所示。

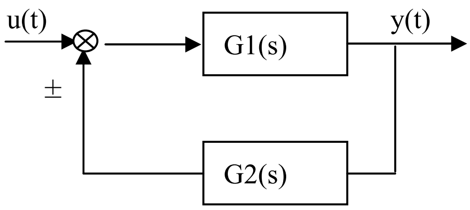


图 6.3 反馈结构

实现反馈结构传递函数的命令：

G=feedback(G1,G2,Sign)

说明：Sign 用来表示正反馈或负反馈，Sign=-1 或省略则表示为负反馈。

【例 6.6】根据系统的结构框图求出整个系统的传递函数，结构框图如图 6.4 所示，其中 $G_1(s)=\frac{1}{s^2+2s+1}$， $G_2(s)=\frac{1}{s+1}$， $G_3(s)=\frac{1}{2s+1}$， $G_4(s)=\frac{1}{s}$。

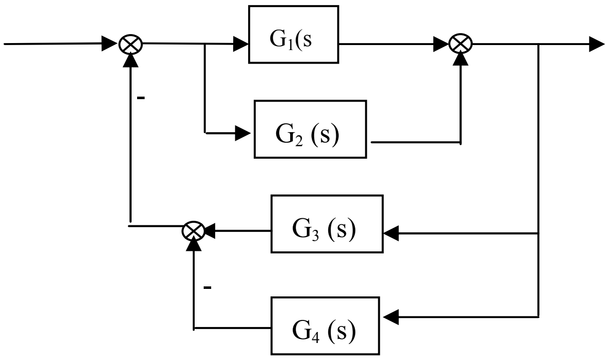


图 6.4 结构框图

```
G1=tf(1,[1 2 1])


Transfer function:
     1
-------------
s^2 + 2 s + 1

G2=tf(1,[1 1]);
G3=tf(1,[2 1]);
G4=tf(1,[1 0]);
G12=G1+G2            %并联结构


Transfer function:
    s^2 + 3 s + 2
---------------------
s^3 + 3 s^2 + 3 s + 1


 G34=G3-G4           %并联结构
```

```
Transfer function:
 -s - 1
---------
2 s^2 + s
```

```
  G=feedback(G12,G34,-1) %反馈结构
```

```
Transfer function:
    2 s^4 + 7 s^3 + 7 s^2 + 2 s
-------------------------------------
2 s^5 + 7 s^4 + 8 s^3 + s^2 - 4 s - 2
```

例如，上图的两个并联结构 G1 和 G2，如果 G1 用状态空间描述，则并联运算的结果也是用状态空间法描述：

```
G1=ss(tf(1,[1 2 1]));          %状态空间描述
G2=tf(1,[1 1]);
G1+G2
```

```
a =
        x1    x2    x3
   x1   -2  -0.25    0
   x2    4     0     0
   x3    0     0    -1


b =
       u1
   x1  0.5
   x2    0
   x3    1


c =
       x1   x2   x3
   y1   0  0.5    1


d =
      u1
   y1  0
```

```
Continuous-time model.
```

4. 复杂的结构框图

求取复杂结构框图的数学模型的步骤：

(1) 将各模块的通路排序编号；

(2) 建立无连接的数学模型：使用 append 命令实现各模块未连接的系统矩阵。

G=append(G1,G2,G3,…)

(3) 指定连接关系：写出各通路的输入输出关系矩阵 Q，第一列是模块通路编号，从第二列开始的几列分别为进入该模块的所有通路编号；INPUTS 变量存储输入信号所加入的通路编号；OUTPUTS 变量存储输出信号所在通路编号。

(4) 使用 connect 命令构造整个系统的模型。

Sys=connect(G,Q,INPUTS,OUTPUTS)

如果各模块都使用传递函数，也可以用 blkbuild 命令建立无连接的数学模型，则第二步修改如下：

将各通路的信息存放在变量中：通路数放在 nblocks，各通路传递函数的分子和分母分别放在不同的变量中；用 blkbuild 命令求取系统的状态方程模型。

【例 6.7】根据图 6.5 所示系统结构框图，求出系统总的传递函数。

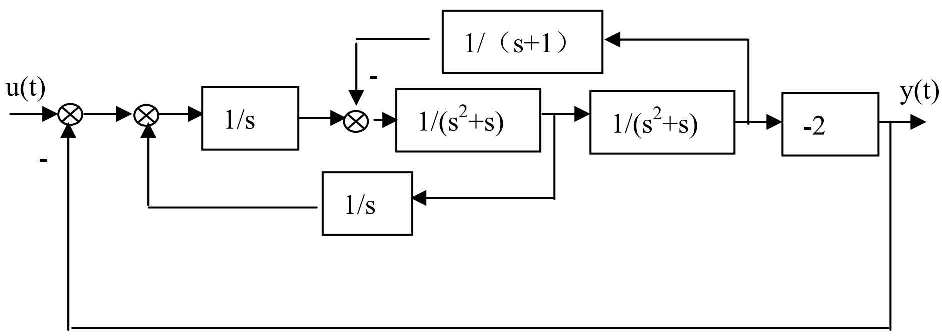


图 6.5 结构框图

方法一：使用 append 命令

(1) 将各模块的通路排序编号，如图 6.6 所示。

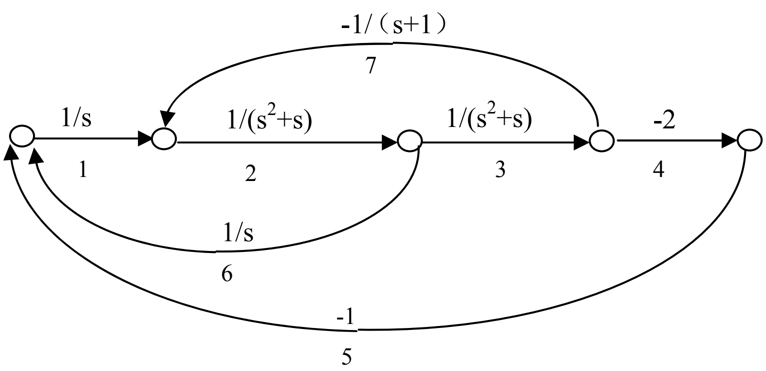


图 6.6 信号流图

(2) 使用 append 命令实现各模块未连接的系统矩阵

```
G1=tf(1,[1 0]);
G2=tf(1,[1 1 0]);
G3=tf(1,[1 1 0]);
```

```
G4=tf(-2,1);
G5=tf(-1,1);
G6=tf(1,[1 0]);
G7=tf(-1,[1 1]);
Sys=append(G1,G2,G3,G4,G5,G6,G7)
```

```
Transfer function from input 1 to output...
      1
 #1:  -
      s

 #2:  0

 #3:  0

 #4:  0

 #5:  0

 #6:  0

 #7:  0

Transfer function from input 2 to output...
 #1:  0

         1
 #2:  -------
      s^2 + s

 #3:  0

 #4:  0

 #5:  0

 #6:  0

 #7:  0

Transfer function from input 3 to output...
 #1:  0

 #2:  0
```

```
        1
 #3:  -------
      s^2 + s

 #4:  0

 #5:  0

 #6:  0

 #7:  0

Transfer function from input 4 to output...
 #1:  0

 #2:  0

 #3:  0

 #4:  -2

 #5:  0

 #6:  0

 #7:  0

Transfer function from input 5 to output...
 #1:  0

 #2:  0

 #3:  0

 #4:  0

 #5:  -1

 #6:  0

 #7:  0

Transfer function from input 6 to output...
```

```
 #1:  0

 #2:  0

 #3:  0

 #4:  0

 #5:  0

      1
 #6:  -
      s

 #7:  0

Transfer function from input 7 to output...
 #1:  0

 #2:  0

 #3:  0

 #4:  0

 #5:  0

 #6:  0

       -1
 #7:  -----
      s + 1
```

程序分析：将每个模块用 append 命令放在一个系统矩阵中，可以看到 Sys 模块存放了七个模块的传递函数，为了节省篇幅在此未列出完整的 Sys 模块。

(3) 指定连接关系

```
 Q=[1 6 5;             %通路 1 的输入信号为通路 6 和通路 5
2 1 7;                   %通路 2 的输入信号为通路 1 和通路 7
3 2 0;                 %通路 3 的输入信号为通路 2
4 3 0;
5 4 0;
6 2 0;
7 3 0;]
 INPUTS=1;         %系统总输入由通路 1 输入
 OUTPUTS=4;        %系统总输出由通路 4 输出
```

```
Q =
     1     6     5
     2     1     7
     3     2     0
     4     3     0
     5     4     0
     6     2     0
     7     3     0
```

程序分析：Q 矩阵建立了各通路之间的关系，共有 7 行；每行的第一列为通路号，从第二列开始为各通路输入信号的通路号；INPUTS 变量存放系统输入信号的通路号；OUTPUTS 变量存放系统输出信号的通路号。

(4) 使用 connect 命令构造整个系统的模型

```
G =connect(Sys,Q,INPUTS,OUTPUTS)


Transfer function:
                   -2 s^2 - 2 s
----------------------------------------------------------
s^7 + 3 s^6 + 3 s^5 + s^4 - s^3 - 3 s^2 - 3 s - 6.661e-016
```

程序分析：用 connect 命令完成整个系统的传递函数模型。

方法二：从第二步开始使用 blkbuild 命令来实现

(2) 将各通路的信息存放在变量中

```
nblocks=7;           %通路数为 7
n1=1; d1=[1 0];             %通路 1 的分子和分母
n2=1; d2=[1 1 0];
n3=1; d3=[1 1 0];
n4=-2; d4=1;
n5=-1; d5=1;
n6=1; d6=[1 0];
n7=-1;d7=[1 1];
```

程序分析：通路数 nblocks 为 7；各通路传递函数的分子存放在变量 ni，分母存放在变量 di。

用 blkbuild 命令求取系统的状态方程模型：

```
blkbuild

State model [a,b,c,d] of the block diagram has 7 inputs and 7 outputs.
```

程序分析：增广状态方程模型即 7 条通路的输入输出信号状态模型建立了，存放在 a、b、c、d 变量中。

(3) 建立连接矩阵 Q 指定连接关系，Q 矩阵同前面

(4) 使用 connect 命令构造整个系统的模型

```
[A,B,C,D]=connect(a,b,c,d,Q,INPUTS,OUTPUTS)

A =
```

```
     0     0     0     0     2     1     0
     1    -1     0     0     0     0    -1
     0     1     0     0     0     0     0
     0     0     1    -1     0     0     0
     0     0     0     1     0     0     0
     0     0     1     0     0     0     0
     0     0     0     0     1     0    -1
B =
     1
     0
     0
     0
     0
     0
     0
C =
     0     0     0     0    -2     0     0
D =
     0
```

6.4 线性系统的时域分析

6.4.1 零输入响应分析

MATLAB 中使用 initial 命令来计算和显示连续系统的零输入响应。

语法：

initial(G,x0, Ts)　　　　　%绘制系统的零输入响应曲线

initial(G1,G2,…,x0, Ts)　　　　%绘制系统多个系统的零输入响应曲线

[y,t,x]=initial(G,x0, Ts)　　%得出零输入响应、时间和状态变量响应

说明：G 为系统模型，必须是状态空间模型；x0 是初始条件；Ts 为时间点，如果是标量则为终止时间，如果是数组，则为计算的时刻，可省略；y 为输出响应；t 为时间向量，可省略；x 为状态变量响应，可省略。

【例 6.8】某反馈系统，前向通道的传递函数为 $G1=\frac{12}{s+4}$，反馈通道传递函数为 $H=\frac{1}{s+3}$，求出其初始条件为[1 2]时的零输入响应，如图 6.7 所示。

```
G1=tf(12,[1 4]);
H=tf(1,[1 3]);
GG=feedback(G1,H)
```

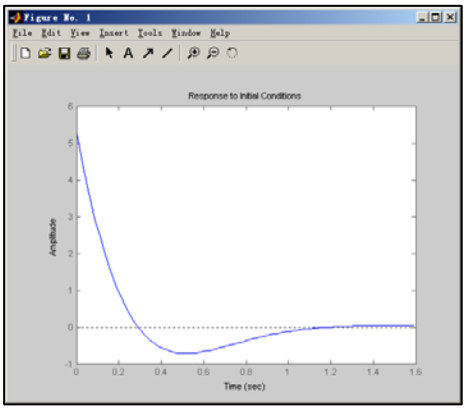


图 6.7 零输入响应曲线

```
Transfer function:
```

```
  12 s + 36
--------------
s^2 + 7 s + 24
```

```
G=ss(GG);
initial(G,[1 2])                %绘制零输入响应
```

2. 离散系统的脉冲响应

离散系统的零输入响应使用 dinitial 命令实现。

语法：

dinitial(a,b,c,d,x0) %绘制离散系统零输入响应

y= dinitial (a,b,c,d,x0) %得出离散系统的零输入响应

[y,x,n]= dinitial (a,b,c,d,x0) %得出离散系统 n 点的零输入响应

说明：a、b、c、d 为状态空间的系数矩阵；x0 为初始条件；y 为输出响应；t 为时间向量；x 为状态变量响应；n 为点数。

6.4.2 脉冲响应分析

1. 连续系统的脉冲响应

连续系统的脉冲响应由 impluse 命令来得出。

语法：

impulse(G, Ts) %绘制系统的脉冲响应曲线

[y,t,x]=impulse(G, Ts) %得出脉冲响应

说明：G 为系统模型，可以是传递函数、状态方程、零极点增益的形式；y 为时间响应；t 为时间向量；x 为状态变量响应，t 和 x 可省略；Ts 为时间点可省略。

【例 6.8 续】求出初始条件为零，该系统的单位脉冲响应并画曲线，如图 6.8 所示。

```
impulse(G)              %绘制脉冲响应曲线
t=0:0.1:10;
y=impulse(G,t)          %根据时间 t 得出脉冲响应
```

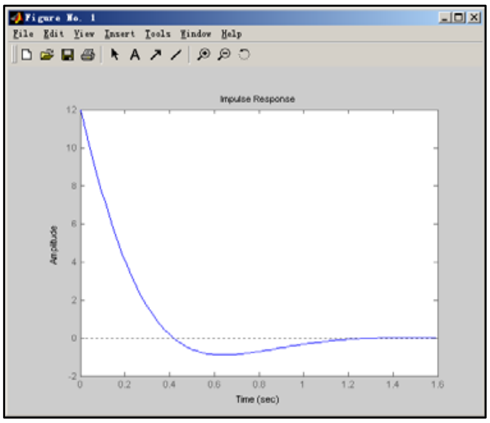


图 6.8 脉冲响应曲线

2. 离散系统的脉冲响应

离散系统的脉冲响应使用 dimpulse 命令实现。

语法：

dimpluse(a,b,c,d,iu) %绘制离散系统脉冲响应曲线

[y,x]=dimpluse(a,b,c,d,iu,n) %得出 n 点离散系统的脉冲响应

[y,x]=dimpluse(num,den,iu,n) %由传递函数得出 n 点离散系统的脉冲响应

说明：iu 为第几个输入信号；n 为要计算脉冲响应的点数；y 的列数与 n 对应；x 为状态变量，可省略。

【例 6.9】根据系统数学模型，得出离散系统的脉冲响应，如图 6.9 所示。

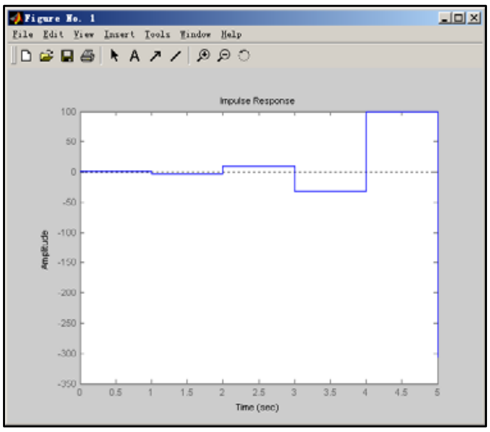


图 6.9 离散系统脉冲响应曲线

```
a=[-2 0;0 -3];
b=[1;1];
c=[1 -4];
d=1;
dimpulse(a,b,c,d,1,10)                  %绘制离散系统脉冲响应的 10 个点
```

6.4.3 阶跃响应分析

1. 连续阶跃响应

阶跃响应可以用 step 命令来实现。

语法：

step(G, Ts) %绘制系统的阶跃响应曲线

[y,t,x]=step(G, Ts) %得出阶跃响应

说明：参数设置与 impulse 命令相同。

【例 6.10】根据【例 6.6】的系统模型得出阶跃响应曲线，如图 6.10 所示。

```
G1=tf(12,[1 4]);
H=tf(1,[1 3]);
G=feedback(G1,H)


Transfer function:
  12 s + 36
--------------
s^2 + 7 s + 24
```

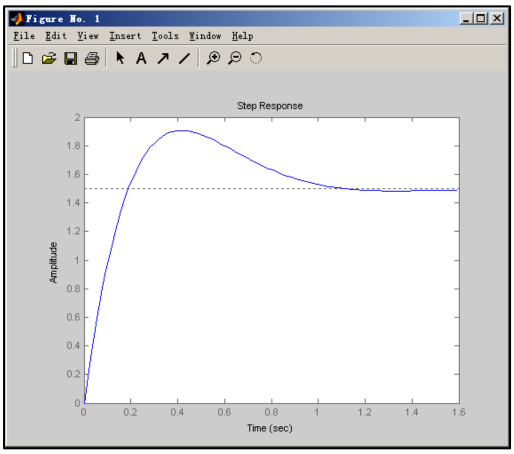


图 6.10 阶跃响应曲线

```
step(G)          %绘制阶跃响应曲线
```

可以由 step 命令根据时间 t 的步长不同，得出不同的阶跃响应波形，如图 6.11 所示。

```
t1=0:0.1:5;
y1=step(G,t1);
plot(t1,y1)
t2=0:0.5:5;
y2=step(G,t2);
plot(t2,y2)
```

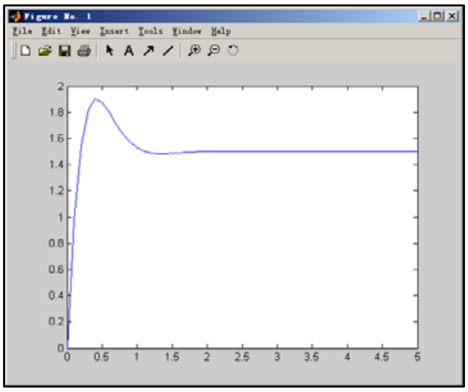
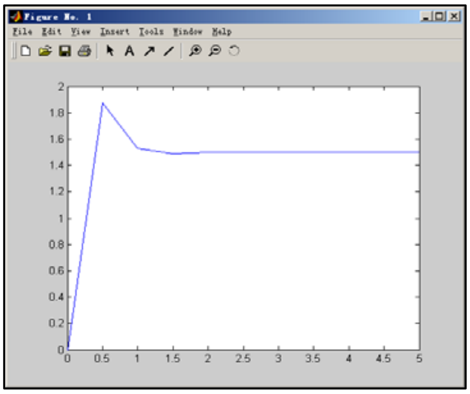


图 6.11 (a) 阶跃响应曲线



(b) 步长增大的阶跃响应曲线

2. 离散系统的阶跃响应

离散系统阶跃响应使用 dstep 命令来实现，语法规则与 dimpluse 相同。

6.4.4 任意输入的响应

1. 连续系统的任意输入响应

连续系统对任意输入的响应用 lsim 命令来实现。

语法：

lsim(G,U,Ts) %绘制系统的任意响应曲线

lsim(G1,G2,…U,Ts) %绘制多个系统任意响应曲线

[y,t,x]=lsim(G,U,Ts) %得出任意响应

说明：U 为输入序列，每一列对应一个输入；Ts 为时间点，U 的行数和 Ts 相对应；参数 t 和 x 可省略。

【例 6.11】根据输入信号和系统的数学模型，得出任意输入的输出响应，输入信号为正

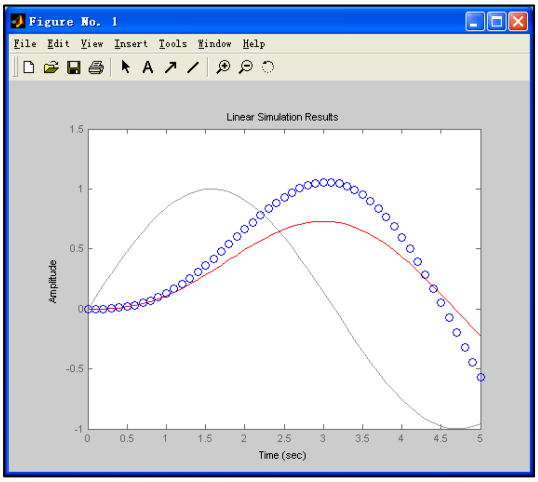


图 6.12 正弦输入信号响应

弦信号，系统为阻尼系数变化的二阶系统，输出响应如图 6.12 所示。

```
t=0:0.1:5;
u=sin(t);
G1=tf(1,[1 1.41 1])


Transfer function:
      1
----------------
s^2 + 1.41 s + 1

 G2=tf(1,[1 0.6 1])


Transfer function:
      1
---------------
s^2 + 0.6 s + 1

lsim(G1,'r',G2,'bo',u,t)          %绘制两个系统的正弦输出响应
```

2. 离散系统的任意输入响应

离散系统的任意输入响应用 dlsim 命令来实现。

语法：

dlsim(a,b,c,d,U)　　　　%绘制离散系统的任意响应曲线

[y,x]=dlsim(num,den,U)　　　　%得出离散系统任意响应和状态变量响应

[y,x]=dlsim(a,b,c,d,U)　　　　%得出离散系统响应和状态变量响应

说明：U 为任意序列输入信号。

【例 6.12】根据离散系统的 Z 变换表达式 $G(z)=\dfrac{2+5z^{-1}+z^{-2}}{1+2z^{-1}+3z^{-2}}$，得出正弦序列输入响应，输出响应如图 6.13 所示。

```
num=[2 5 1];
den=[1 2 3];
t=0:0.1:5;
u=sin(t);
y=dlsim(num,den,u);
```

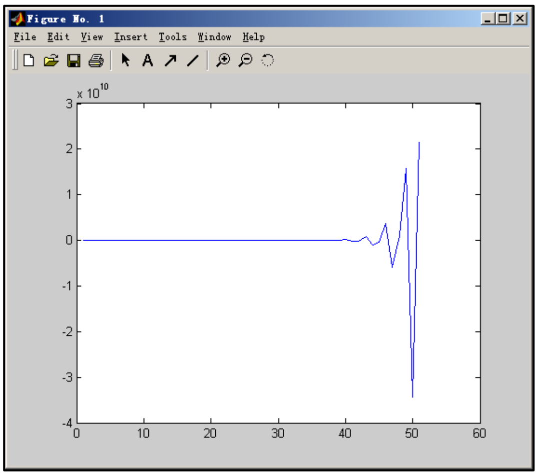


图 6.13 正弦输入信号响应

6.4.5 系统的结构参数

1. 极点和零点

(1) pole 命令计算极点

语法：

p=pole(G)

说明：当系统有重极点时，计算结果不一定准确。

(2) tzero 命令计算零点和增益

语法：

z=tzero(G)　　　　　　%得出连续和离散系统的零点

[z,gain]=tzero(G)　　　　%获得零点和零极点增益

说明：对于单输入单输出系统， tzero 命令也用来计算零极点增益。

【例 6.13】获得 $G(s)=\dfrac{5s+100}{s^4+8s^3+32s^2+80s+100}$ 系统的零极点。

```
num=[5 100];
den=[1 8 32 80 100];
G=tf(num,den);
p=pole(G)

p =
  -1.0000 + 3.0000i
  -1.0000 - 3.0000i
  -3.0000 + 1.0000i
  -3.0000 - 1.0000i
[z,gain]=tzero(G)        %得出零点和零极点增益

z =
   -20
gain =
     5
```

2. 闭环系统的阻尼系数和固有频率

damp 命令用来计算闭环系统所有共轭极点的阻尼系数ζ和固有频率ωn。

语法：

[wn,zeta]=damp(G)

3. 时域响应的稳态增益

稳态增益可使用 dcgain 命令来得出。

语法：

k=dcgain(G)　　　　　　%获得稳态增益

【例 6.13 续】计算所有闭环极点的ζ和ωn。

```
[wn,zeta]=damp(G)

wn =
```

```
    3.1623
    3.1623
    3.1623
    3.1623
zeta =
    0.9487
    0.9487
    0.3162
    0.3162
 dcgain(G)                        %得出线性系统的稳态增益

ans =
     1
```

6.5 线性系统的频域分析

6.5.1 频域特性

频域特性由下式求出：

Gw=polyval(num,j*w)./polyval(den,j*w)

mag=abs(Gw) %幅频特性

pha=angle(Gw) %相频特性

说明：j 为虚部变量。

【例 6.14】由二阶系统传递函数 $G(s)=\dfrac{1}{s^2+1.414s+1}$，得出频域特性。

```
 num=1;
 den=[1 1.414 1];
 w=1 ;
 Gw=polyval(num,j*w)./polyval(den,j*w)           %得出系统频率特性

Gw =
        0 - 0.7072i
 Aw=abs(Gw)                                %得出幅频特性

Aw =
    0.7072
 Fw=angle(Gw)                                  %得出相频特性

Fw =
   -1.5708
```

6.5.2 连续系统频域特性

1. bode 图

bode 图是对数幅频和对数相频特性曲线。

语法：

bode(G,w)	%绘制 bode 图
[mag,pha]=bode(G,w)	%得出 w 对应的幅值和相角
[mag,pha,w]=bode(G)	%得出幅值、相角和频率

说明：G 为系统模型，w 为频率向量，mag 为系统的幅值，pha 为系统的相角。

【例 6.15】根据系统传递函数 $G(s)=\dfrac{1}{s(s+1)(s+2)}$，绘制 bode 图如图 6.14(a)所示。

```
num=1;
den=conv([1 1],[1 ,2])

den =
     1     3     2
G=tf(num,[den 0])


Transfer function:
       1
-----------------
s^3 + 3 s^2 + 2 s

bode(G)                      %绘制 bode 图
```

【例 6.15 续】使用 semilogx 命令绘制对数幅频和相频特性，如图 6.14(b)所示。

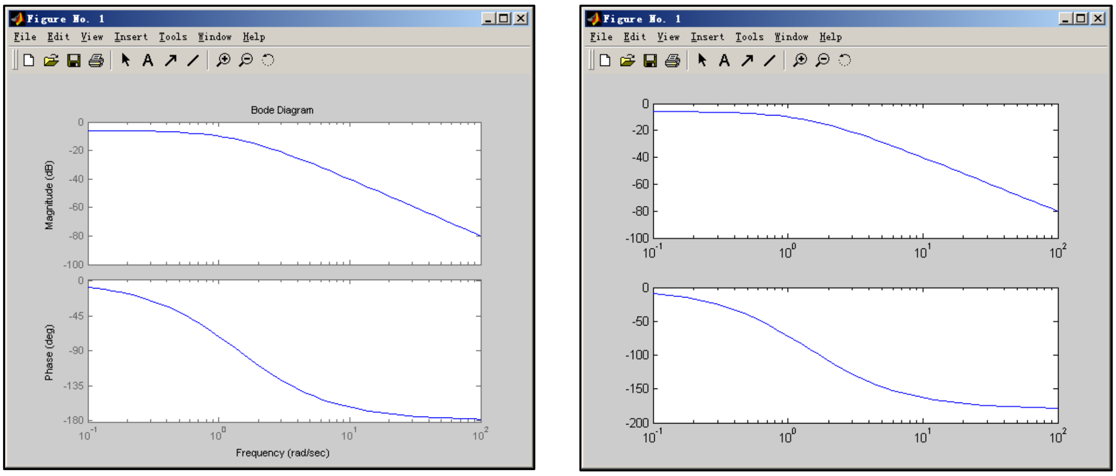


图 6.14 (a) Bode 图　　　　(b) 用 semilogx 命令绘制对数幅频和相频特性

```
w=logspace(-1,2);
[m,p]=bode(num,den,w);
subplot(2,1,1)
semilogx(w,20*log10(m))
subplot(2,1,2)
```

```
semilogx(w,p)
```

2. nyquist 曲线

nyquist 曲线是幅相频率特性曲线，使用 nyquist 命令绘制和计算。

语法：

nyquist (G,w)	%绘制 nyquist 曲线
nyquist (G1,G2,…w)	%绘制多条 nyquist 曲线
[Re,Im]= nyquist (G,w)	%由 w 得出对应的实部和虚部
[Re,Im,w]= nyquist (G)	%得出实部、虚部和频率

说明：G 为系统模型；w 为频率向量，也可以用{wmin,wmax}表示频率的范围；Re 为频率特性的实部，Im 为频率特性的虚部。

【例 6.16】 根据传递函数 $G1(s)=\dfrac{1}{s(s+1)(s+2)}$、$G2(s)=\dfrac{1}{(s+1)(s+2)}$ 和 $G3(s)=\dfrac{1}{s(s+1)}$，绘制各系统的 nyquist 曲线，如图 6.15 所示。

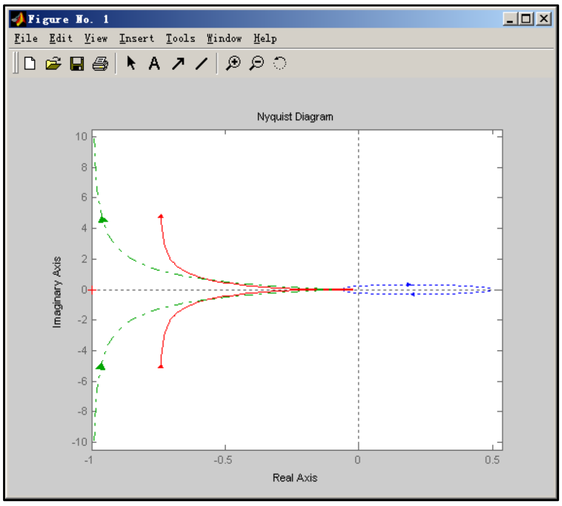


图 6.15 nyquist 曲线

```
num=1;
den1=[conv([1 1],[1 2]),0];
G1=tf(num,den1)


Transfer function:
       1
-----------------
s^3 + 3 s^2 + 2 s

den2=[conv([1 1],[1 2])];
G2=tf(num,den2)


Transfer function:
     1
-------------
s^2 + 3 s + 2

den3=[1 1 0];
G3=tf(num,den3)


Transfer function:
   1
-------
```

```
s^2 + s
```

```
nyquist(G1,'r',G2,'b:',G3,'g-.',{0.1,180/57.3})    % 频 率 范 围
{0.1,180/57.3}
```

获得频率特性的实部和虚部：

```
w=1:2;
[re,im]=nyquist(G1,w)
```

```
re(:,:,1) =
   -0.3000
re(:,:,2) =
   -0.0750
im(:,:,1) =
   -0.1000
im(:,:,2) =
    0.0250
```

程序分析：re 和 im 是三维数组，组成为(Ny, Nu, Length(w))，其中 Ny 为输出，Nu 为输入。

3. nichols 图

nichols 图是对数幅相频率特性曲线，使用 nichols 命令绘制和计算。

语法：

nichols (G,w)	%绘制 nichols 图
nichols (G1,G2,…w)	%绘制多条 nichols 图
[Mag,Pha]= nichols (G,w)	%由 w 得出对应的幅值和相角
[Mag,Pha,w]= nichols (G)	%得出幅值、相角和频率

在单位反馈系统中，闭环系统的传递函数可以写成 G(s)/(1+G(s))，因此 nichols 图的等 M 圆和等 N 圆就映射成等 M 线和等 α 线，MATLAB 提供了绘制 nichols 框架下的等 M 线和等 α 线的命令 ngrid。

语法：

ngrid(‘new’)	%清除图形窗口并绘制等 M 线和等 α 线

说明：‘new’为创建的图形窗口，清除该图形窗口并绘制等 M 线和等 α 线，如果绘制了 nichols 图后可省略‘new’，直接添加等 M 线和等 α 线；产生-40db～40db 的幅值和–360°～0° 的范围，并保持图形。

【例 6.16 续】根据传递函数 $G1(s)=\dfrac{1}{s(s+1)(s+2)}$，绘制等 M 线等 α 线和 nichols 图，如图 6.16 所示。

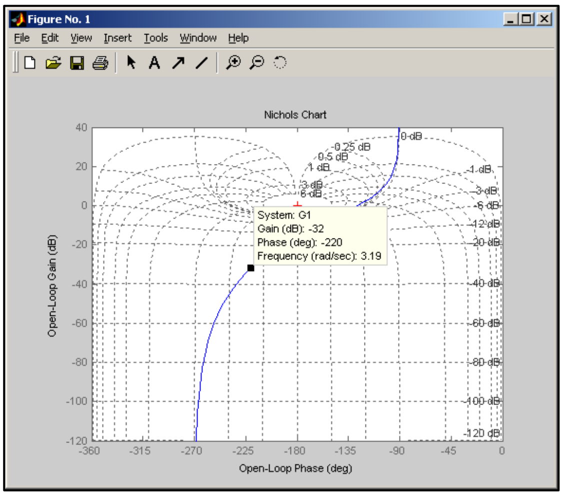


图 6.16 nichols 图

```
ngrid('nichols1')              %绘制等 M 线和等 α 线
nichols(G1)                 %绘制 nichols 图
w=1:2;
[Mag,Pha]=nichols(G1,w)        %获得幅值和相角数值
```

```
Mag(:,:,1) =
    0.3162
Mag(:,:,2) =
    0.0791
Pha(:,:,1) =
 -161.5651
Pha(:,:,2) =
 -198.4349
```

程序分析：用“ngrid”命令可以创建等 M 线和等 α 线图形窗口，用鼠标单击图形中的某点，可以看到其相关信息。

6.5.3 幅值裕度和相角裕度

语法：

margin(G)　　　　%绘制 bode 图并标出幅值裕度和相角裕度

[Gm,Pm,Wcg,Wcp]=margin(G)　　　　%得出幅值裕度和相角裕度

说明：Gm 为幅值裕度，Wcg 为幅值裕度对应的频率；Pm 为相角裕度，Wcp 为相角裕度对应的频率(穿越频率)。如果 Wcg 或 Wcp 为 nan 为 Inf，则对应的 Gm 或 Pm 为无穷大。

【例 6.16 续】得出 $G1(s)=\dfrac{1}{s(s+1)(s+2)}$ 系统的幅值裕度和相角裕度。

```
G1
```

```
Transfer function:
```

```
        1
-----------------
s^3 + 3 s^2 + 2 s
```

```
[Gm,Pm,Wcg,Wcp]=margin(G1)
```

```
Gm =
    6.0000
Pm =
   53.4109
Wcg =
    1.4142
Wcp =
    0.4457
```

6.5.4 离散系统频域分析

【例 6.17】绘制 $G(z)=\dfrac{2+5z^{-1}+z^{-2}}{1+2z^{-1}+3z^{-2}}$ 系统的 bode 图，如图 6.17 所示。

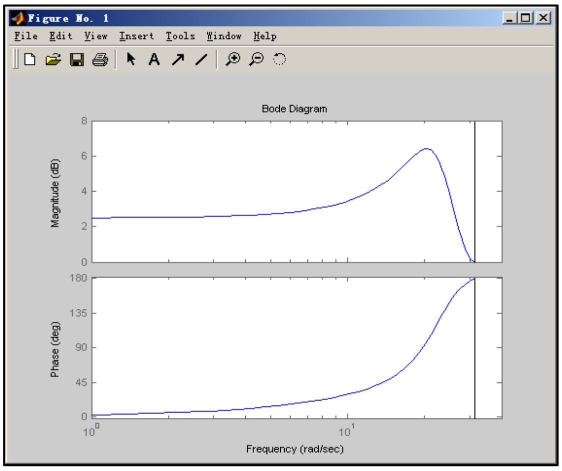


图 6.17 离散系统的 bode 图

```
dnum=[2 5 1];
dden=[1 2 3];
dbode(dnum,dden,0.1)              %绘制 bode 图，采样周期为 0.1s
```

6.6 线性系统的根轨迹分析

6.6.1 绘制根轨迹

MATLAB 中绘制根轨迹使用 rlocus 命令。

语法：

```
rlocus(G)                %绘制根轨迹
rlocus(G1,G2,…)          %绘制多个系统的根轨迹
[r,k]=rlocus(G)          %得出闭环极点和对应的 K
r= rlocus(G,k)           %根据 K 得出对应的闭环极点
```

1. 常规根轨迹

【例 6.18】绘制开环传递函数 $G(s)=\frac{k}{s(s+4)(s+2-4j)(s+2+4j)}$ 的根轨迹，如图 6.18 所示。

```
num=1 ;
den=[conv([1,4],conv([1 -2+4i],[1 -2-4i])),0]

den =
    1     0     4    80     0
G=tf(num,den)


Transfer function:
       1
------------------
s^4 + 4 s^2 + 80 s

rlocus(G)                %绘制根轨迹
[r,k]=rlocus(G);            %得出闭环极点和增益
```

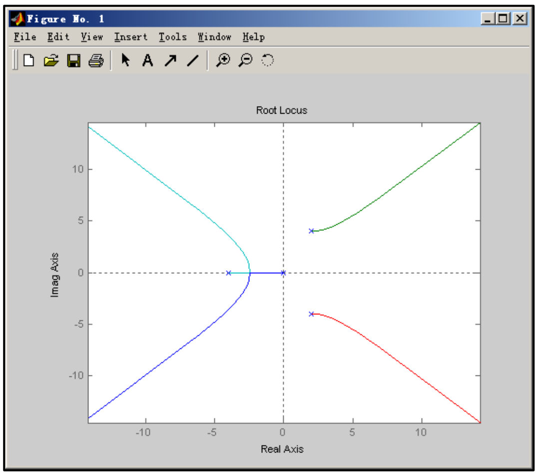


图 6.18 根轨迹

2. 零度根轨迹

由于是正反馈，其闭环特征方程为：1-G(s)=0，因此将分子多项式取负号就可以了。

【例 6.19】系统前向通道传递函数为 $G(s)=\frac{k(s+2)}{(s+3)(s^2+2s+2)}$ 的正反馈，绘制其根轨迹，如图 6.19 所示。

```
num=[-1 -2];
den=conv([1 3],[1 2 2]);
G=tf(num,den)
```

```
Transfer function:
      -s - 2
---------------------
s^3 + 5 s^2 + 8 s + 6
```

```
rlocus(G)
```

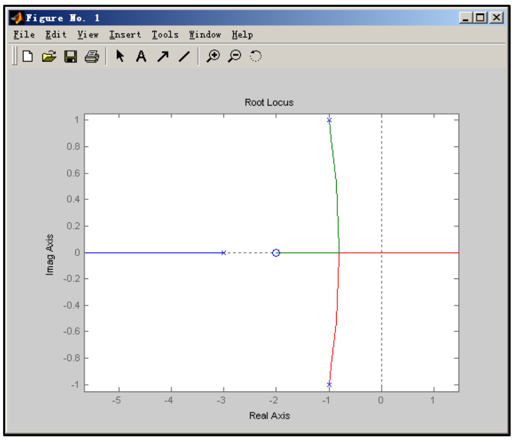


图 6.19 零度根轨迹

6.6.2 根轨迹的其它工具

1. 指定点的开环增益

MATLAB 控制系统工具箱提供了 rlocfind 命令，可以在绘制的根轨迹上获得定位点的增益 K 值。

语法：

[k,p]=rlocfind(G) %获得定位点的增益和极点

说明：k 和 p 分别是定位点的增益和极点。

该命令在产生根轨迹后执行，执行该命令后，在命令窗口会出现提示"Select a point in the graphics window"，鼠标在图形窗口显示为十字形，当单击根轨迹上的某点时就会获得该点的增益 k 和对应的所有极点 p。

【例 6.19 续】在上图 6.19 中使用 rlocfind 命令。

```
[k,p]=rlocfind(G)
Select a point in the graphics window
selected_point =
 -8.2323e-001 -1.5326e-001i
k =
   1.8558e+000
p =
 -3.3843e+000
 -8.0785e-001 +1.5352e-001i
 -8.0785e-001 -1.5352e-001i
```

程序分析：在绘制的根轨迹图中，单击鼠标可以获得该点的坐标，以及增益 k 和对应的所有极点 p。

2. 主导极点的等ζ线和等ωn 线

语法：

sgrid('new') %清除图形窗口绘制等ζ线和等ωn 线

sgrid(zeta,wn,'new') %绘制指定的等ζ线和等ωn 线

说明：'new'为创建的新图形窗口，清除该图形窗口并绘制等ζ线和等ωn 线，如果绘制了根轨迹图后可省略‘new’，直接添加等 M 线和等α线；zeta 和 wn 分别为指定的ζ和ωn。

【例 6.20】绘制开环传递函数为 $G1(s)=\frac{k}{s(s+1)(s+2)}$ 的系统根轨迹，如图 6.20 所示，并找出ζ=0.707 附近的点，绘制出其相应的阶跃响应曲线。

```
num=1;
den=[conv([1 1],[1 2 ]),0];
G1=tf(num,den);
rlocus(G1)                    %在上图中绘制根轨迹
sgrid(0.707,10)               %绘制ζ=0.707 线和ωn=10 线
```

在等ζ线和等ωn 线图中ζ=0.707 的线上，取根轨迹点的增益，将该增益构成闭环传递函数，画出其阶跃响应曲线。

```
 [k,p]=rlocfind(G1)           %获取鼠标单击点的增益和所有极点
Select a point in the graphics window
selected_point =
   -0.3791 + 0.3602i
k =
      0.6233
p =
   -2.2279
   -0.3861 + 0.3616i
   -0.3861 - 0.3616i
 G=feedback(k*G1,1)           %得出闭环传递函数
Transfer function:
              0.6233
--------------------------
s^3 + 3 s^2 + 2 s + 0.6233
 figure(2)
 step(G)                      %绘制阶跃响应曲线
```

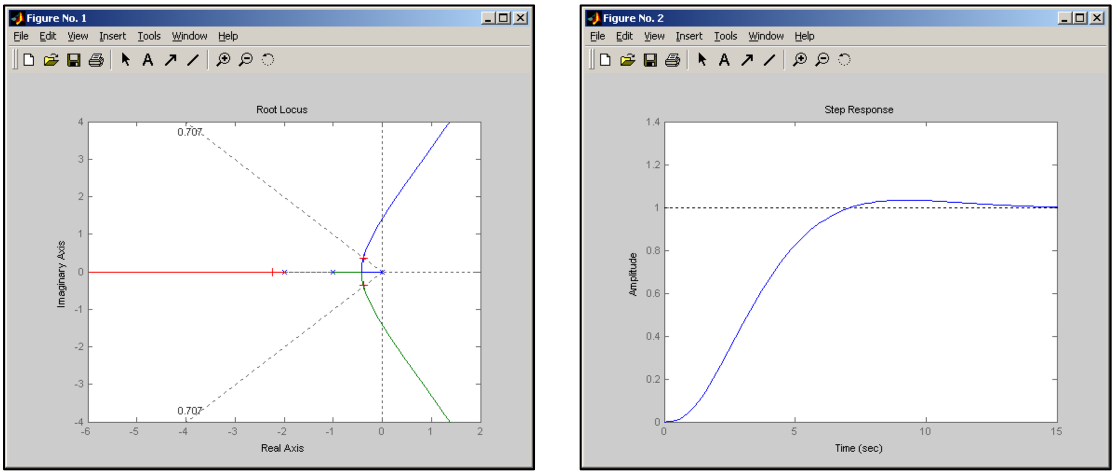


图 6.20 (a) 等ζ线和等ω n 线　　　　(b) 阶跃响应曲线

程序分析：可以看出其阶跃响应的性能较好，根据鼠标单击处的系统参数绘制阶跃响应曲线。

3. 系统根轨迹的设计工具 RLTool

MATLAB 控制工具箱还提供了一个系统根轨迹分析的图形界面，使用 rltool 命令打开该界面。

语法：

rltool　　　　%打开空白的根轨迹分析的图形界面

rltool(G)　　　　%打开某系统根轨迹分析的图形界面

【例 6.21】将开环传递函数 $\frac{k}{s(s+4)(s^2+4s+20)}$ 的根轨迹用系统根轨迹分析的图形界面来分析。

```
num=1;
a=[1 0];
b=[1 4];
c=[1 4 20];
den=conv(a,b);
den=conv(den,c);
G=tf(num,den);
rltool(G);
```

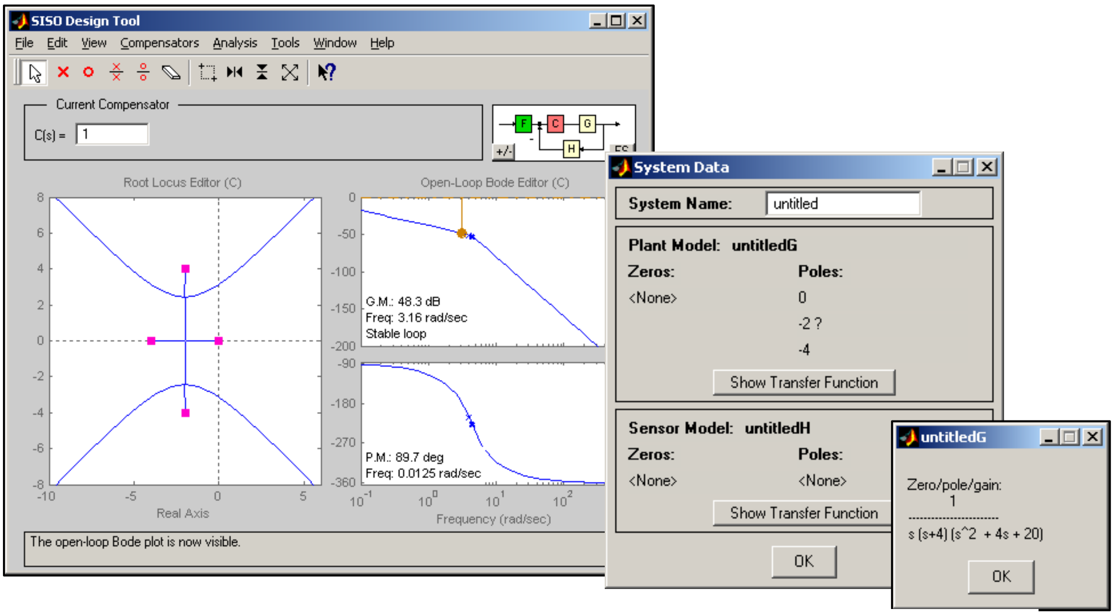


图 **6.21** 系统根轨迹分析的图形界面

在命令窗口输入“sisotool(G)”命令时，也可以打开如上图所示的图形界面。

6.7 线性系统的状态空间设计

6.7.1 极点配置法

1. 单输入系统的极点配置

MATLAB 使用 acker 命令来对单输入单输出系统极点配置。

语法：

k=acker(A,B,p) %SISO 系统极点配置

说明：A、B 为系统矩阵；p 为期望特征值数组。

【例 6.22】已知系统状态方程 $\dot{x} = Ax(t) + Bu(t)$，$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -5 & -6 \end{bmatrix}$，$B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$，期望特征值为 p=[-2+2j,-2-2j,-10]，求状态增益矩阵 k。

```
A=[0 1 0;0 0 1;-1 -5 -6];
B=[0;0;1];
p=[-2+2j -2-2j -10];
k=acker(A,B,p)
```

```
k =
    79    43     8
```

2. 多输入系统的极点配置

3. 离散系统的极点配置

6.7.2 最优二次型设计

1.　连续系统最优二次型设计

MATLAB 使用 lqr 命令来求解最优问题。

语法：

[K,P,E]=lqr(A,B,Q,R)　　　　%连续系统最优二次型调节器设计

说明：A、B、Q、R 矩阵定义如上，P 为 Riccati 方程的解，K 为最优增益反馈矩阵，E 为闭环特征值。

【例 6.23】 已知系统状态方程 $\dot{x} = Ax(t) + Bu(t)$，$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -35 & -27 & -9 \end{bmatrix}$，$B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$，

$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$，R=1，求最优二次型解。

```
a=[0 1 0;0 0 1;-35 -27 -9];
b=[0;0;1];
q=eye(3);
r=1;
[k,p,e]=lqr(a,b,q,r)
```

```
k =
    0.0143    0.1107    0.0676
p =
    4.2625    2.4957    0.0143
    2.4957    2.8150    0.1107
    0.0143    0.1107    0.0676
e =
  -5.0958
  -1.9859 + 1.7110i
  -1.9859 - 1.7110i
```

程序分析：得出最优反馈增益矩阵 k，系统闭环特征值 e 和 Riccati 方程的正定矩阵解。

2.　离散系统最优二次型设计

MATLAB 提供的离散最优设计的命令为 dlqr。

语法：

[K,P,E]=dlqr(A,B,Q,R)　　　　%离散最优二次型调节器设计

3. 对输出加权的最优二次型设计

在很多情况下，需要对输出量加权而不是对状态量加权，其代价函数：

$$J = \frac{1}{2}\int_0^{\infty} [y(t)^T Qy(t) + u(t)^T Ru(t)]dt$$

MATLAB 使用 lqry 命令解相应的 Riccati 方程和最优反馈增益。

语法：

[K,P,E]=lqry(A,B,Q,R)　　　　%系统加权最优二次型调节器设计

第 7 章　Simulink 仿真环境

Simulink 是面向框图的仿真软件。

7.1 演示一个 Simulink 的简单程序

【例 7.1】创建一个正弦信号的仿真模型。

步骤如下：

(1) 在 MATLAB 的命令窗口运行 simulink 命令，或单击工具栏中的图标，就可以打开 Simulink 模块库浏览器(Simulink Library Browser) 窗口，如图 7.1 所示。

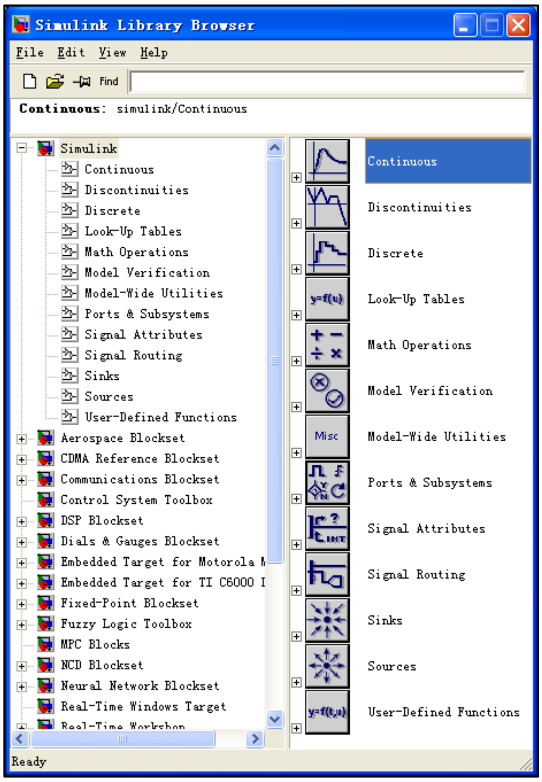


图 7.1 Simulink 界面

(2) 单击工具栏上的图标或选择菜单“File”——“New”——“Model”，新建一个名为“untitled”的空白模型窗口。

(3) 在上图的右侧子模块窗口中，单击“Source”子模块库前的“+”(或双击 Source)，或者直接在左侧模块和工具箱栏单击 Simulink 下的 Source 子模块库，便可看到各种输入源模块。

(4) 用鼠标单击所需要的输入信号源模块“Sine Wave”(正弦信号)，将其拖放到的空白模型窗口“untitled”，则“Sine Wave”模块就被添加到 untitled 窗口；也可以用鼠标选中“Sine Wave”模块，单击鼠标右键，在快捷菜单中选择“add to 'untitled'”命令，就可以将“Sine Wave”模块添加到 untitled 窗口，如图 7.2 所示。

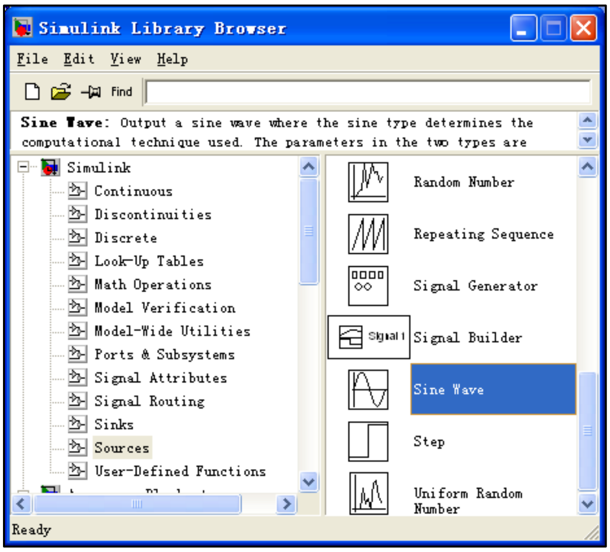


图 7.2 Simulink 界面

(5) 用同样的方法打开接收模块库“Sinks”，选择其中的“Scope”模块(示波器)拖放到“untitled”窗口中。

(6) 在“untitled”窗口中，用鼠标指向“Sine Wave”右侧的输出端，当光标变为十字符时，按住鼠标拖向“Scope”模块的输入端，松开鼠标按键，就完成了两个模块间的信号线连接，一个简单模型已经建成。如图 7.3 所示。

(7) 开始仿真，单击“untitled”模型窗口中“开始仿真”图标，或者选择菜单“Simulink”——“Start”，则仿真开始。双击“Scope”模块出现示波器显示屏，可以看到黄色的正弦波形。如图 7.4 所示。

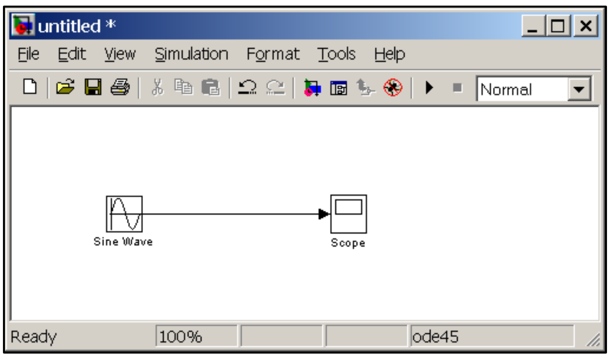


图 7.3 Simulink 模型窗口

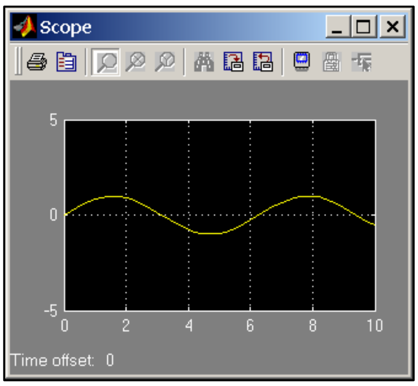


图 7.4 示波器窗口

(8) 保存模型，单击工具栏的图标，将该模型保

存为“Ex0701.mdl”文件。

7.2 Simulink 的文件操作和模型窗口

7.2.1 Simulink 的文件操作

1. 新建文件

新建仿真模型文件有几种操作：

- 在 MATLAB 的命令窗口选择菜单“File”→“New”→“Model”。
- 在图 7.1 的 Simulink 模块库浏览器窗口选择菜单“File”→“New”→“Model”，或者单击工具栏的图标。
- 在图 7.3 的 Simulink 模型窗口选择菜单“File”→“New”→“Model”，或者单击工具栏的图标。

2. 打开文件

打开仿真模型文件有几种操作：

- 在 MATLAB 的命令窗口输入不加扩展名的文件名，该文件必须在当前搜索路径中，例如输入“Ex0701”。
- 在 MATLAB 的命令窗口选择菜单“File”→Open…”或者单击工具栏的图标打开文件。
- 在图 7.1 的 Simulink 模块库浏览器窗口选择菜单“File”→“Open…”或者单击工具栏的图标打开“.mdl”文件。
- 在图 7.3 的 Simulink 模型窗口中选择菜单“File”→“Open…”或者单击工具栏的图标打开文件。

7.2.2 Simulink 的模型窗口

模型窗口由菜单、工具栏、模型浏览器窗口、模型框图窗口以及状态栏组成。

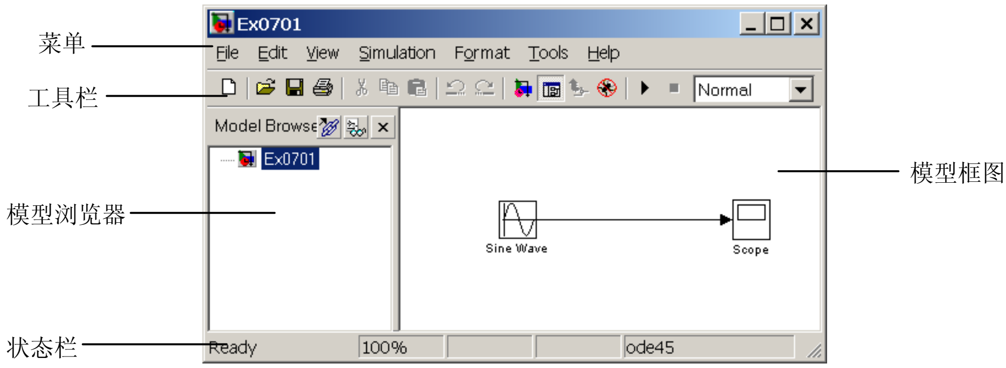


图 7.5 双窗口模型窗口

1. 状态栏

2. 工具栏

模型窗口工具栏如图 7.6 所示。

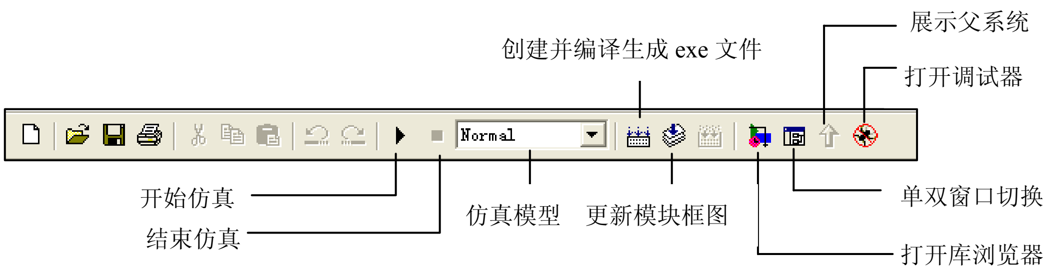


图 7.6 工具栏

3. 菜单

Simulink 的模型窗口的常用菜单如表 7.1 所示。

表 7.1 模型窗口常用菜单表

菜单名	菜单项	功能
File	New——Model	新建模型
	Model properties	模型属性
	Preferences	SIMULINK 界面的默认设置选项
	Print…	打印模型
	Close	关闭当前 Simulink 窗口
	Exit MATLAB	退出 MATLAB 系统
Edit	Create subsystem	创建子系统
	Mask subsystem…	封装子系统
	Look under mask	查看封装子系统的内部结构
	Update diagram	更新模型框图的外观
View	Go to parent	显示当前系统的父系统
	Model browser options	模型浏览器设置
	Block data tips options	鼠标位于模块上方时显示模块内部数据
	Library browser	显示库浏览器

	Fit system to view	自动选择最合适的显示比例
	Normal	以正常比例(100%)显示模型
Simulation	Start / Stop	启动 / 停止仿真
	Pause / Continue	暂停 / 继续仿真
	Simulation Parameters…	设置仿真参数
	Normal	普通 Simulink 模型
	Accelerator	产生加速 Simulink 模型
Format	Text alignment	标注文字对齐工具
	Filp name	翻转模块名
	Show / Hide name	显示 / 隐藏模块名
	Filp block	翻转模块
	Rotate Block	旋转模块
	Library link display	显示库链接
	Show / Hide drop shadow	显示 / 隐藏阴影效果
	Sample time colors	设置不同的采样时间序列的颜色
	Wide nonscalar lines	粗线表示多信号构成的向量信号线
	Signal dimensions	注明向量信号线的信号数
	Port data types	标明端口数据的类型
	Storage class	显示存储类型
Tools	Data explorer…	数据浏览器
	Simulink debugger…	Simulink 调试器
	Data class designer	用户定义数据类型设计器
	Linear Analysis	线性化分析工具

7.3 模型的创建

7.3.1 模块的操作

1. 对象的选定

- 选定单个对象

选定对象只要在对象上单击鼠标，被选定的对象的四角处会出现小黑块编辑框。

- 选定多个对象

如果选定多个对象，可以按下 Shift 键，然后再单击所需选定的模块；或者用鼠标拉出矩形虚线框，将所有待选模块框在其中，则矩形框中所有的对象均被选中，如图 7.7 所示。

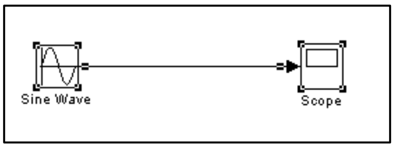


图 7.7 选定多个对象

- 选定所有对象

如果要选定所有对象，可以选择菜单“Edit”→“Select all”。

2. 模块的复制

(1) 不同模型窗口(包括模型库窗口)之间的模块复制

- 选定模块，用鼠标将其拖到另一模型窗口。
- 选定模块，使用菜单的“Copy”和“Paste”命令。
- 选定模块，使用工具栏的“Copy”和“Paste”按钮。

(2) 在同一模型窗口内的复制模块(如图 7.8 所示)

- 选定模块，按下鼠标右键，拖动模块到合适的地方，释放鼠标。
- 选定模块，按住 Ctrl 键，再用鼠标拖动对象到合适的地方，释放鼠标。
- 使用菜单和工具栏中的“Copy”和“Paste”按钮。

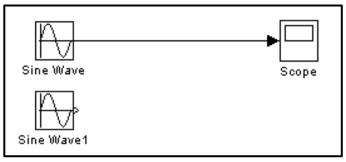


图 7.8 在同一模型窗口复制对象

3. 模块的移动

- 在同一模型窗口移动模块

选定需要移动模块，用鼠标将模块拖到合适的地方。

- 在不同模型窗之间移动模块

在不同模型窗之间移动模块，在用鼠标移动的同时按下 Shift 键。

当模块移动时，与之相连的连线也随之移动。

4. 模块的删除

要删除模块，应选定待删除模块，按 Delete 键；或者用菜单“Edit”→“Clear”或“Cut”；或者用工具栏的“Cut”按钮。

5. 改变模块大小

选定需要改变大小的模块，出现小黑块编辑框后，用鼠标拖动编辑框，可以实现放大或缩小。

6. 模块的翻转

- 模块翻转 180 度

选定模块，选择菜单“Format”→“Flip Block”可以将模块旋转 180 度，如同 7.9 中间为翻转 180 度示波器模块。

▪ 模块翻转 90 度

选定模块，选择菜单“Format”→“Rotate　Block”可以将模块旋转 90 度，如图 7.9 右边示波器所示。如果一次翻转不能达到要求，可以多次翻转来实现。

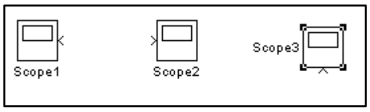


图 7.9 翻转模块

7. 模块名的编辑

▪ 修改模块名

单击模块下面或旁边的模块名，出现虚线编辑框就可对模块名进行修改。

▪ 模块名字体设置

选定模块，选择菜单“Format”→“Font”，打开字体对话框设置字体。

▪ 模块名的显示和隐藏

选定模块，选择菜单“Format”→“Hide /Show name”，可以隐藏或显示模块名。

▪ 模块名的翻转

选定模块，选择菜单“Format”→“Flip name”，可以翻转模块名。

7.3.2 信号线的操作

1. 模块间连线

先将光标指向一个模块的输出端，待光标变为十字符后，按下鼠标键并拖动，直到另一模块的输入端。

2. 信号线的分支和折曲

(1) 分支的产生

将光标指向信号线的分支点上，按鼠标右键，光标变为十字符，拖动鼠标直到分支线的终点，释放鼠标；或者按住 Ctrl 键，同时按下鼠标左键拖动鼠标到分支线的终点，如图 7.10 所示。

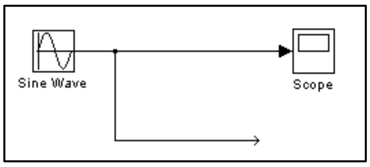


图 7.10 信号线的分支

(2) 信号线的折线

选中已存在的信号线，将光标指向折点处，按住 Shift 键，同时按下鼠标左键，当光标变成小圆圈时，用鼠标拖动小圆圈将折点拉至合适处，释放鼠标，如图 7.11 所示。

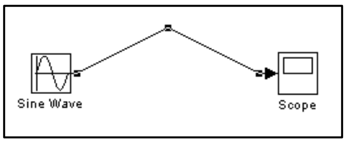


图 7.11 信号线的折线

3. 信号线文本注释(label)

- 添加文本注释

双击需要添加文本注释的信号线，则出现一个空的文字填写框，在其中输入文本。

- 修改文本注释

单击需要修改的文本注释，出现虚线编辑框即可修改文本。

- 移动文本注释

单击标识，出现编辑框后，就可以移动编辑框。

- 复制文本注释

单击需要复制的文本注释，按下 Ctrl 键同时移动文本注释，或者用菜单和工具栏的复制操作。

4. 在信号线中插入模块

如果模块只有一个输入端口和一个输出端口，则该模块可以直接被插入到一条信号线中。

7.3.3 给模型添加文本注释

(1) 添加模型的文本注释

在需要当作注释区的中心位置，双击鼠标左键，就会出现编辑框，在编辑框中就可以输入文字注释。

(2) 注释的移动

在注释文字处单击鼠标左键，当出现文本编辑框后，用鼠标就可以拖动该文本编辑框。

7.4 Simulink 的基本模块

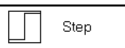
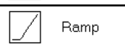
7.4.1 基本模块

Simulink 的基本模块包括 9 个子模块库。

1. 输入信号源模块库(Sources)

输入信号源模块是用来向模型提供输入信号。常用的输入信号源模块源如表 7.2 所示。

表 7.2 常用的输入信号源模块表

名称	模块形状	功能说明
Constant		恒值常数，可设置数值
Step		阶跃信号
Ramp		线性增加或减小的信号

Sine Wave	Sine Wave	正弦波输出
Signal Generator	Signal Generator	信号发生器，可以产生正弦、方波、锯齿波和随机波信号
From File	untitled.mat From File	从文件获取数据
From Workspace	simin From Workspace	从当前工作空间定义的矩阵读数据
Clock	Clock	仿真时钟，输出每个仿真步点的时间
In	1 In1	输入模块

2. 接收模块库(Sinks)

接收模块是用来接收模块信号的，常用的接收模块如表 7.3 所示。

表 7.3 常用的接收模块表

名称	模块形状	功能说明
Scope	Scope	示波器，显示实时信号
Display	Display	实时数值显示
XY Graph	XY Graph	显示 X-Y 两个信号的关系图
To File	untitled.mat To File	把数据保存为文件
To Workspace	simout To Workspace	把数据写成矩阵输出到工作空间
Stop Simulation	STOP Stop Simulation	输入不为零时终止仿真，常与关系模块配合使用
Out	1 Out1	输出模块

3. 连续系统模块库(Continuous)

连续系统模块是构成连续系统的环节，常用的连续系统模块如表 7.4 所示。

表 7.4 常用的连续系统模块表

名称	模块形状	功能说明
Integrator	1/s Integrator	积分环节
Derivative	du/dt Derivative	微分环节
State-Space	x' = Ax+Bu y = Cx+Du State-Space	状态方程模型
Transfer Fcn	1/(s+1) Transfer Fcn	传递函数模型
Zero-Pole	(s-1)/(s(s+1)) Zero-Pole	零—极点增益模型
Transport Delay	Transport Delay	把输入信号按给定的时间做延时

4. 离散系统模块库(Discrete)

离散系统模块是用来构成离散系统的环节，常用的离散系统模块如表 7.5 所示。

表 7.5 常用的离散系统模块表

名称	模块形状	功能说明
Discrete Transfer Fcn	1/(z+0.5) Discrete Transfer Fcn	离散传递函数模型
Discrete Zero-Pole	(z-1)/(z(z-0.5)) Discrete Zero-Pole	离散零极点增益模型
Discrete State-Space	Discrete State-Space	离散状态方程模型
Discrete Filter	1/(1+0.5z-1) Discrete Filter	离散滤波器
Zero-Order Hold	Zero-Order Hold	零阶保持器

First-Order Hold	First-Order Hold	一阶保持器
Unit Delay	$\frac{1}{z}$ Unit Delay	采样保持，延迟一个周期

7.4.2 常用模块的参数和属性设置

1. 模块参数设置

(1) 正弦信号源(Sine Wave)

双击正弦信号源模块，会出现如图 7.13 所示的参数设置对话框。

图 7.13 的上部分为参数说明，仔细阅读可以帮助用户设置参数。Sine type 为正弦类型，包括 Time-based 和 Sample-based；Amplitude 为正弦幅值；Bias 为幅值偏移值；Frequency 为正弦频率；Phrase 为初始相角；Sample time 为采样时间。

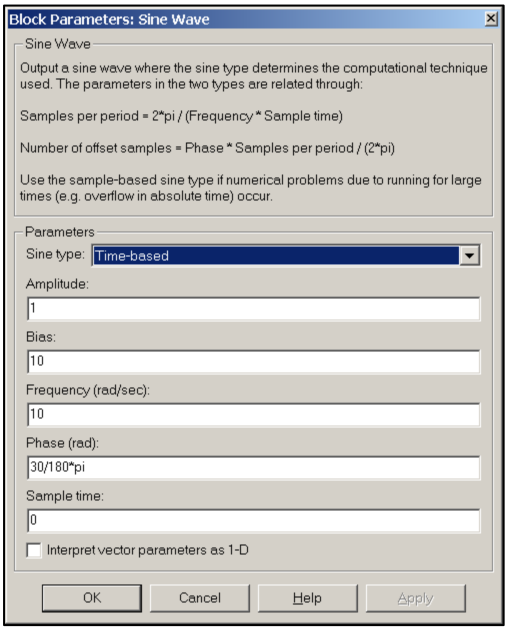


图 7.13 模块的参数设置

(2) 阶跃信号源(Step)

阶跃信号模块是输入信号源，其模块参数对话框如图 7.14 所示。

其中：Step time 为阶跃信号的变化时刻，initial value 为初始值，Final value 为终止值，Sample time 为采样时间。

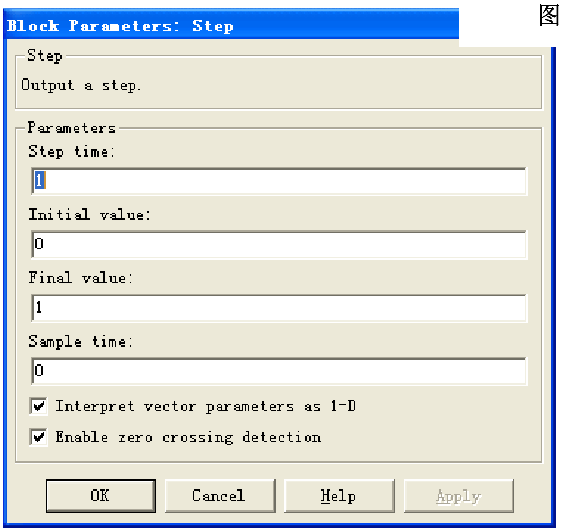


图 7.14 阶跃信号模块的参数

(3) 从工作空间获取数据(From workspace)

从工作空间获取数据模块的输入信号源为工作空间。

【例 7.2】在工作空间计算变量 t 和 y，将其运算的结果作为系统的输入。

```
t=0:0.1:10;
y=sin(t);
t=t';
y=y';
```

然后将“From Workspace”模块的参数设置对话框打开，如图 7.15(a)所示，在“Data”栏填写“[t,y]”，单击“OK”按钮完成。则在模型窗口中该模块就显示为图 7.15(b)。用示波器作为接收模块，可以查看输出波形为正弦波。

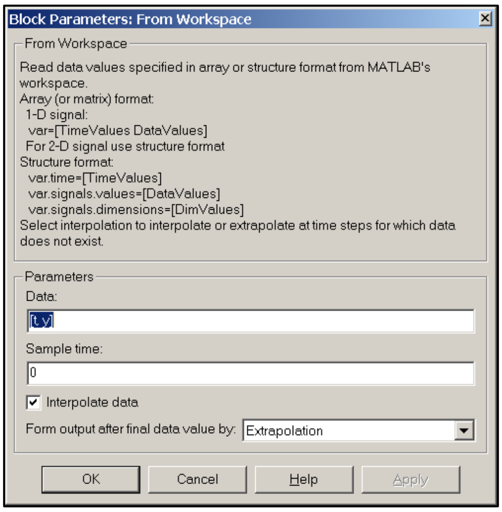




图 7.15 (a) 模块参数设置　　　　(b) 从工作空间获取数据模块

“Data”的输入有几种，可以是矩阵、包含时间数据的结构数组。“From Workspace”模块的接收模块必须有输入端口，“Data”矩阵的列数应等于输入端口的个数+1，第一列自动当成时间向量，后面几列依次对应各端口。

```
t=0:0.1:2*pi;
y=sin(t);
y1=[t;y];
save Ex0702 y1                    %保存在“Ex0702.mat”文件中
```

(4) 从文件获取数据(From file)

从文件获取数据模块是指从 mat 数据文件中获取数据为系统的输入。

将【例 7.2】中的数据保存到.mat 文件：

```
t=0:0.1:2*pi;
y=sin(t);
y1=[t;y];
save Ex0702 y1              %保存在“Ex0702.mat”文件中
```

然后将“From File”模块的参数设置对话框打开，如图 7.16 所示，在“File name”栏填写“Ex0702.mat”，单击“OK”按钮完成。用示波器作为接收模块，可以查看输出波形。

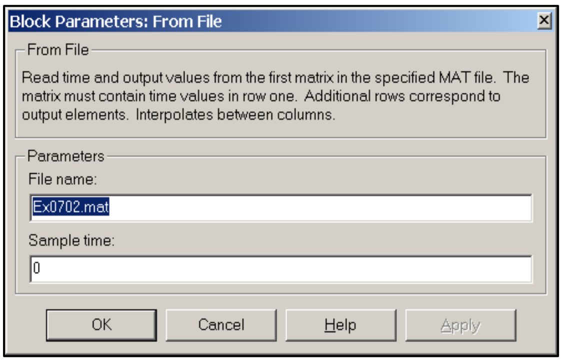


图 7.16 From File 参数设置

(5) 传递函数(Transfer function)

传递函数模块是用来构成连续系统结构的

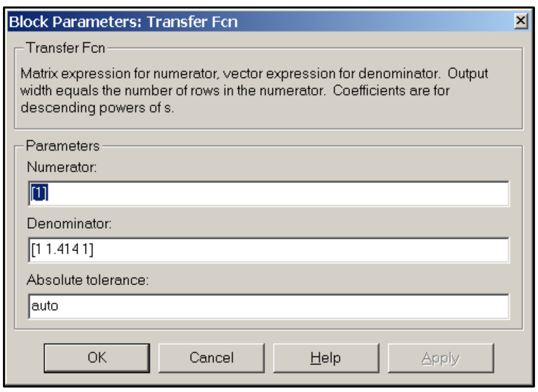


图 7.17 传递函数模块参数设置

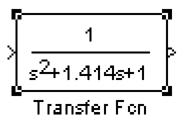


图 7.18

模块，其模块参数对话框如图 7.17 所示。

在上图中设置“Denominator”为“[1 1.414 1]”，则在模型窗口中显示为如图 7.18 所示。

(6) 示波器(Scope)

示波器模块是用来接收输入信号并实时显示信号波形曲线，示波器窗口的工具栏可以调整显示的波形，显示正弦信号的示波器如图 7.19 所示。

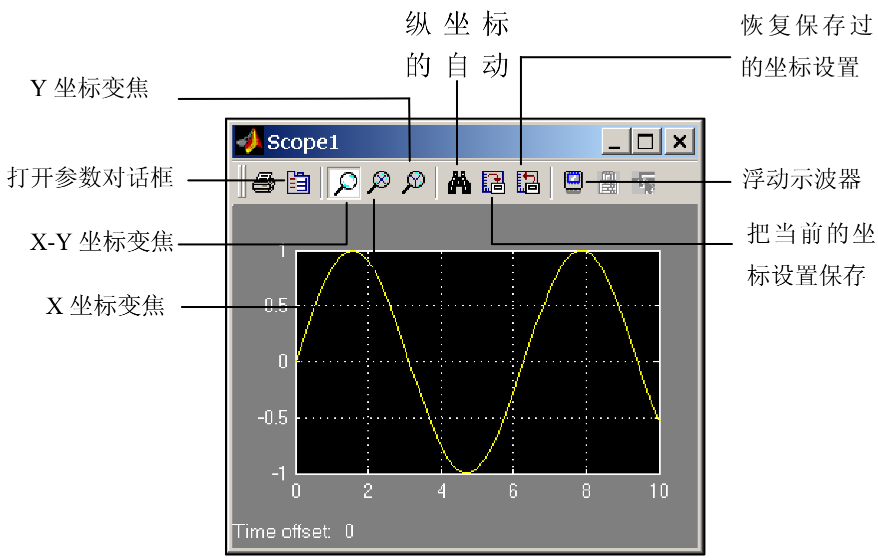


图 7.19 示波器窗口

2. 模块属性设置

每个模块的属性对话框的内容都相同，如图 7.22 所示。

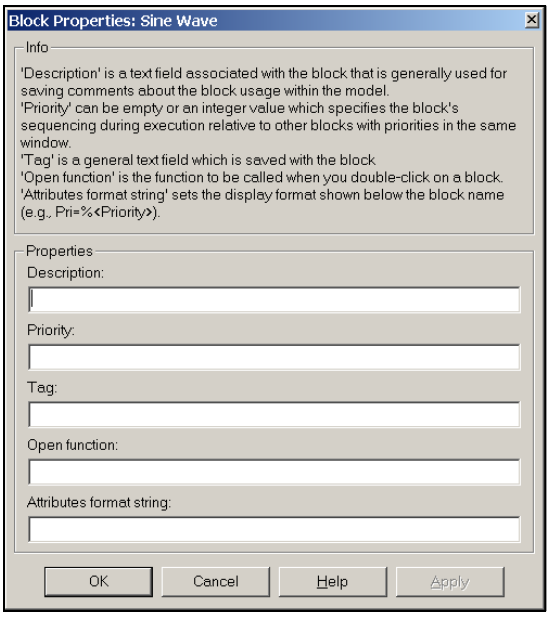


图 7.22 模块的属性设置

(1) 说明(Description)

对模块在模型中用法的注释。

(2) 优先级(Priority)

规定该模块在模型中相对于其它模块执行的优先顺序。

(3) 标记(Tag)

用户为模块添加的文本格式标记。

(4) 调用函数(Open function)

当用户双击该模块时调用的 MATLAB 函数。

(5) 属性格式字符串(Attributes format string)

指定在该模块的图标下显示模块的哪个参数和格式。

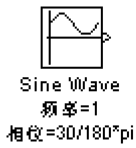


图 7.23 模块的属性格式字符串

7.5 复杂系统的仿真与分析

Simulink 的模型实际上是定义了仿真系统的微分或差分方程组，而仿真则是用数值解算法来求解方程。

7.5.1 仿真的设置

在模型窗口选择菜单“Simulation”→“Simulation parameters…”，则会打开参数设置对话框，如图 7.24 所示。

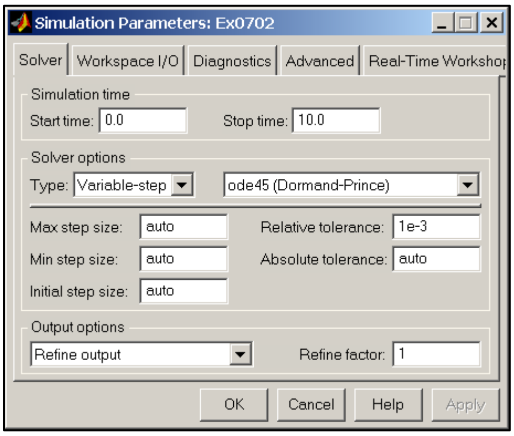


图 7.24 Solver 参数设置

1. Solver 页的参数设置

(1) 仿真的起始和结束时间

仿真的起始时间(Start time)

仿真的结束时间(Stop time)

(2) 仿真步长

仿真的过程一般是求解微分方程组，“Solve options”的内容是针对解微分方程组的设置。

(3) 仿真解法

Type 的右边：设置仿真解法的具体算法类型。

(4) 输出模式

根据需要选择输出模式(Output options)，可以达到不同的输出效果。

2. Workspace I/O(工作空间输入输出)页的设置

如图 7.25 所示，可以设置 Simulink 从工作空间输入数据、初始化状态模块，也可以把仿真的结果、状态模块数据保存到当前工作空间。

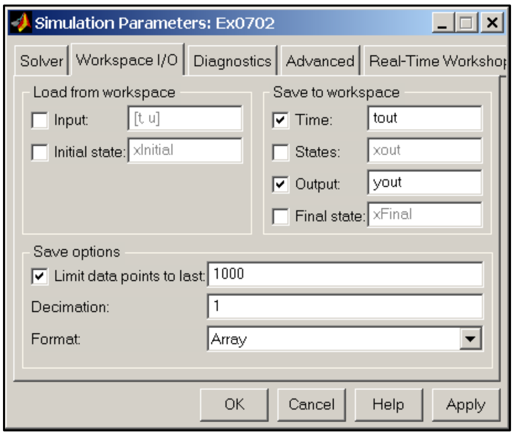


图 7.25 Workspace I/O 参数设置

(1) 从工作空间装载数据(Load from workspace)

(2) 保存数据到工作空间(Save to workspace)

- Time 栏

勾选 Time 栏后，模型将把(时间)变量以在右边空白栏填写的变量名(默认名为 tout)存放于工作空间。

- States 栏

勾选 States 栏后，模型将把其状态变量在右边空白栏填写的变量名(默认名为 xout)存放于工作空间。

- Output 栏

如果模型窗口中使用输出模块“Out”，那么就必须勾选 Output 栏，并填写在工作空间中的输出数据变量名(默认名为 yout)。

- Final state 栏

Final state 栏的勾选，将向工作空间以在右边空白栏填写的名称(默认名为 xFinal)，存放最终状态值。

(3) 变量存放选项(Save options)

Save options 必须与 Save to workspace 配合使用。

7.5.2 连续系统仿真

【例 7.3】建立二阶系统的仿真模型。

方法一：

输入信号源使用阶跃信号，系统使用开环传递函数 $\frac{1}{s^2+0.6s}$，接受模块使用示波器来构成模型。

(1) 在“Sources”模块库选择“Step”模块，在“Continuous”模块库选择“Transfer Fcn”模块，在“Math Operations”模块库选择“Sum”模块，在“Sinks”模块库选择“Scope”。

(2) 连接各模块，从信号线引出分支点，构成闭环系统。

(3) 设置模块参数，打开“Sum”模块参数设置对话框，如图 7.26 所示。将“Icon shape”设置为“rectangular”，将“List of signs”设置为“|+-”，其中“|”表示上面的入口为空。

“Transfer Fcn”模块的参数设置对话框中，将分母多项式“Denominator”设置为“[1 0.6

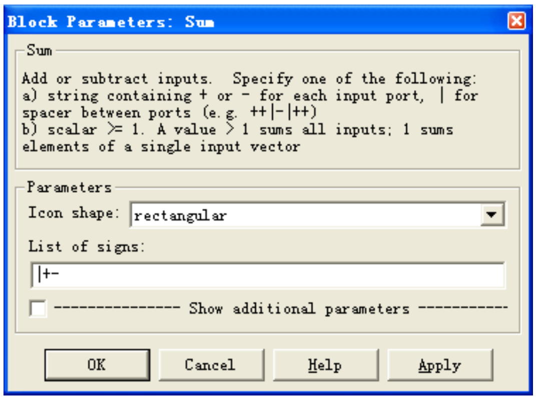


图 7.26 Sum 参数设置

0]”。

将“Step”模块的参数设置对话框中，将“Step time”修改为 0。

(4) 添加信号线文本注释

双击信号线，出现编辑框后，就输入文本。则模型如图 7.27 所示。

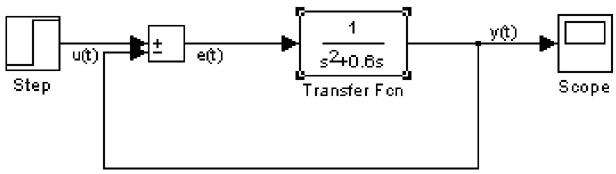


图 7.27 二阶系统模型

(5) 仿真并分析

单击工具栏的“Start simulation”按钮，开始仿真，在示波器上就显示出阶跃响应。

在 Simulink 模型窗口，选择菜单“Simulation”——“Simulation parameters…”命令，在“Solver”页将“Stop time”设置为 15，然后单击“Start simulation”按钮，示波器显示的就到 15 秒结束。

打开示波器的 Y 坐标设置对话框，将 Y 坐标的“Y-min”改为 0，“Y-max”改为 2，将“Title”设置为“二阶系统时域响应”，则示波器如图 7.28 所示。

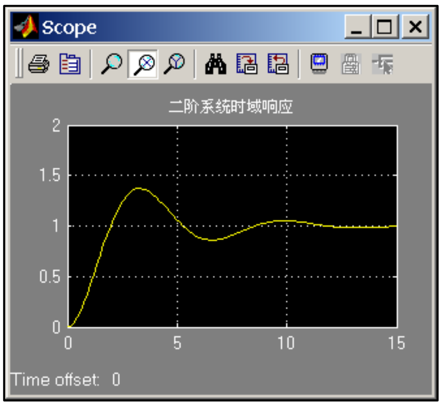


图 7.28 示波器显示

方法二：

(1) 系统使用积分模块(Integrator)和零极点模块(zero-pole)串联，反馈使用“Math Operations”模块库中的“Gain”模块构成反馈环的增益为-1。

(2) 连接模块，由于“Gain”模块在反馈环中，因此需要使用“Flip Block”翻转该模块。

(3) 设置模块参数，将“zero-pole”模块参数对话框中的“Zeros”栏改为“[]”，将“Poles”栏改为[-0.6]。

将“Gain”模块的“Gain”参数改为-1。模型如图 7.29 所示。

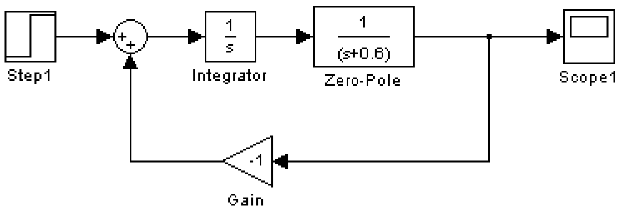


图 7.29 二阶系统模型

如果将示波器换成“Sinks”模块库中的“Out”模块 1 Out1；然后在仿真参数设置对话框的“Workspace I/O” 页(工作空间输入输出)，将“Time”和“Output”栏勾选，并分别设置保存在工作空间的时间量和输出变量为“tout”和“yout”。仿真后在工作空间就可以使用这两个变量来绘制曲线，如图 7.30 所示：

```
plot(tout,yout)
```

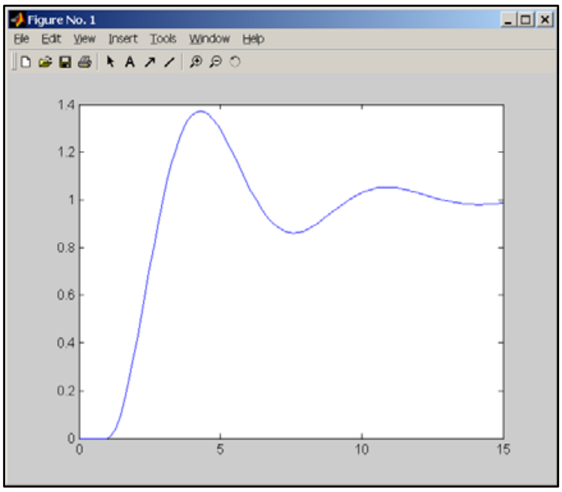


图 7.30　plot 绘制的时域响应波形

7.5.3 离散系统仿真

【例 7.4】控制部分为离散环节，被控对象为两个连续环节，其中一个有反馈环，反馈环引入了零阶保持器，输入为阶跃信号。

创建模型并仿真：

(1) 选择一个“Step”模块，选择两个“Transfer Fcn”模块，选择两个“Sum”模块，选择两个“Scope”模块，选择一个“Gain”模块，在“Discrete”模块库选择一个“Discrete Filter”和一个“Zero-Order Hold”模块。

(2) 连接模块，将反馈环的“Gain”模块和“Zero-Order Hold”模块翻转。

(3) 设置参数，“Discrete Filter”和“Zero-Order Hold”模块的“Sample time”都设置为 0.1s。

(4) 添加文本注释，系统框图如图 7.31 所示。

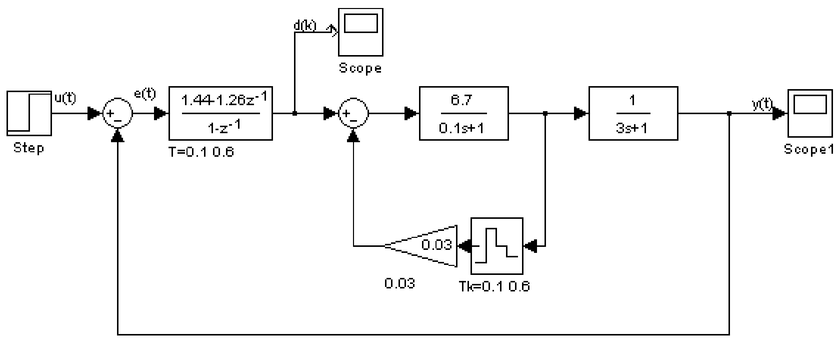


图 7.31 离散系统框图

(5) 设置颜色，Simulink 为帮助用户方便地跟踪不同采样频率的运作范围和信号流向，可以采用不同的颜色表示不同的采样频率，选择菜单“Format”→“Sample time color”，就可以看到不同采样频率的模块颜色不同。

(6) 开始仿真，在 Simulink 模型窗口，选择菜单“Simulation”→“Simulation parameters…”，将“Max step size”设置为 0.05s，则两个示波器“Scope”和“Scope1”的显示如图 7.32 所示。

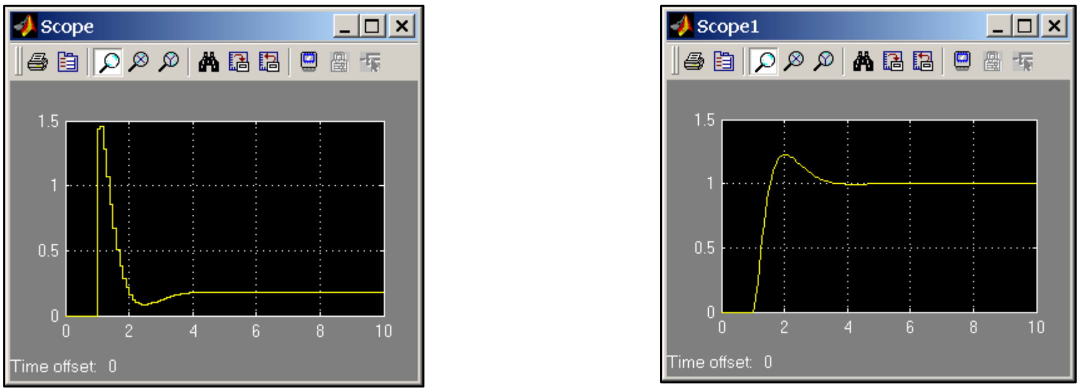


图 7.32 T=Tk=0.1 (a) d(k)示波器显示 (b) y(t)示波器显示

可以看出当 T=Tk=0.1 时系统的输出响应较平稳。

(7) 修改参数，将“Discrete Filter”模块的“Sample time”设置为 0.6s，“Zero-Order Hold”模块的“Sample time”不变；选择菜单“Edit”→“Update diagram”命令修改颜色，就可以看到“Discrete Filter”模块的颜色变化了；然后开始仿真，则示波器显示如图 7.33 所示。

可以看出当 T=0.6 而 Tk=0.1 时，系统出现振荡。

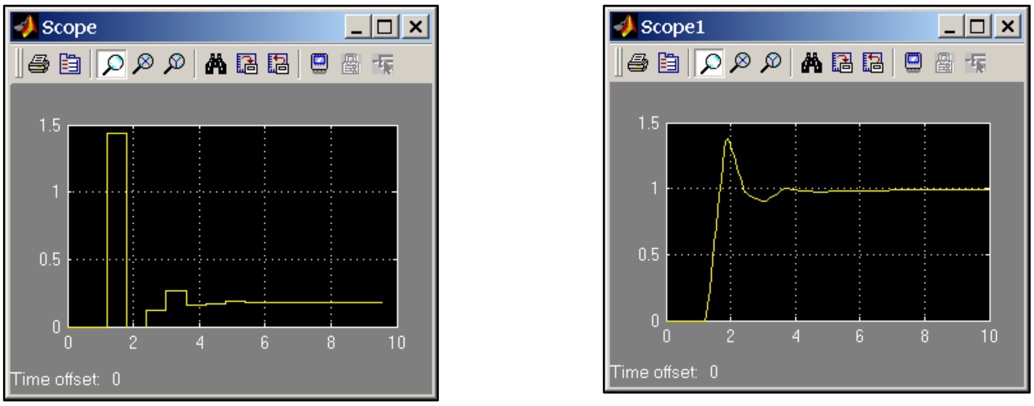


图 7.33 T=0.6 Tk=0.1 (a) d(k)示波器显示 (b) y(t)示波器显示

(8) 修改参数，将“Discrete Filter” 和“Zero-Order Hold”模块的“Sample time”都设置为 0.6s，更新框图颜色，开始仿真，则示波器显示如图 7.34 所示。

当 T=Tk=0.6 时，系统出现强烈的振荡。

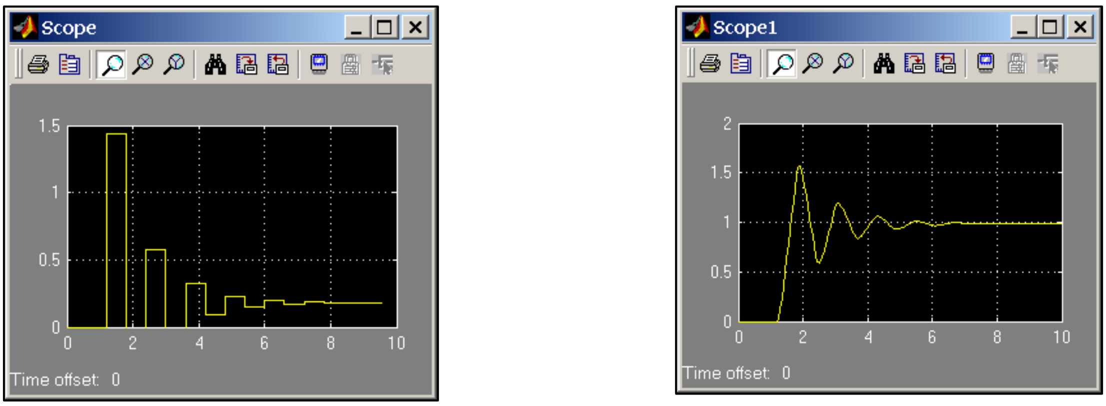


图 7.34 T=0.6 Tk=0.1 (a) d(k)示波器显示 (b) y(t)示波器显示

7.5.4 仿真结构参数化

当系统参数需要经常改变或由函数得出时，可以使用变量来作为模块的参数。

【例 7.5】将【例 7.4】中的模块结构参数用变量表示，结构图如图 7.35 所示。

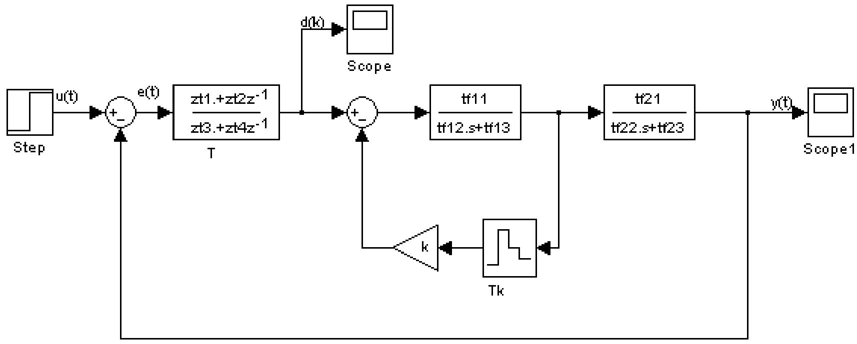


图 7.35 离散系统框图

将参数设置放在 Ex0705_1.m 文件中：

```
% Ex0705_1        参数设置
T=0.1;                               %控制环节采样时间
Tk=0.6;                              %零阶保持器采样时间
k=0.03;                              %Gain 增益
zt1=1.44;zt2=-1.26;zt3=1;zt4=-1;
tf11=6.7;tf12=0.1;tf13=1;
tf21=1;tf22=3;tf23=1
```

在 MATLAB 工作空间运行该文件：

```
Ex0705_1
```

7.6 子系统与封装

7.6.1 建立子系统

子系统类似于编程语言中的子函数。建立子系统有两种方法：在模型中新建子系统和在已有的子系统基础上建立。

1. 在已建立的模型中新建子系统

【例 7.6】打开【例 7.4】建立的模型，将控制对象中的第一个连续环节中的反馈环建立为一个子系统。

在模型窗口中，将控制对象中的第一个连续环节的反馈环用鼠标拖出的虚线框框住，选择菜单“Edit”→“Create subsystem”，则系统如图 7.36 所示。

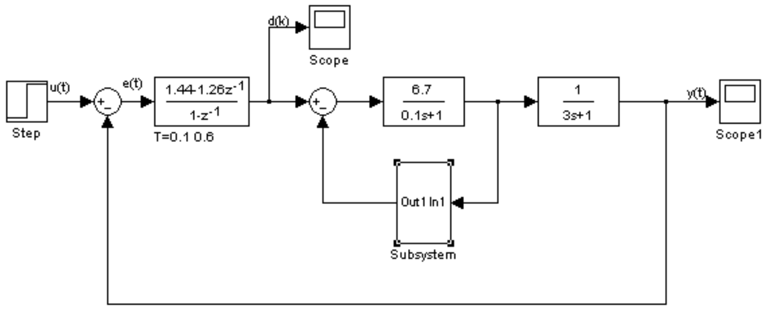


图 7.36 子系统建立

双击子系统，则会出现“Subsystem”模型窗口，如图 7.37 所示。可以看到子系统模型除了用鼠标框住的两个环节，还自动添加了一个输入模块“In1”和一个输出模块“Out1”。

2. 在已有的子系统基础上建立

【例 7.7】在【例 7.6】的基础上建立新子系统，将【例 7.6】模型的控制对象中的第一个对象环节整个作为一个子系统。

将图 7.36 中的所有对象都复制到新的空白模型窗口中，双击打开子系统“Subsystem”，则出现如图 7.37 所示的子系统模型窗口，添加模型构成反馈环形成闭环系统，如图 7.38 所示。

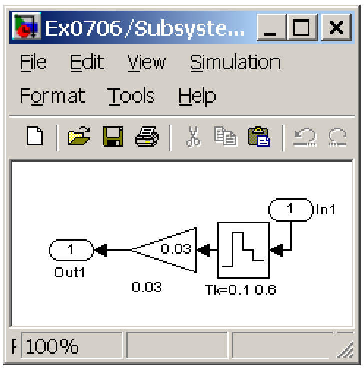


图 7.37 子系统模型窗口

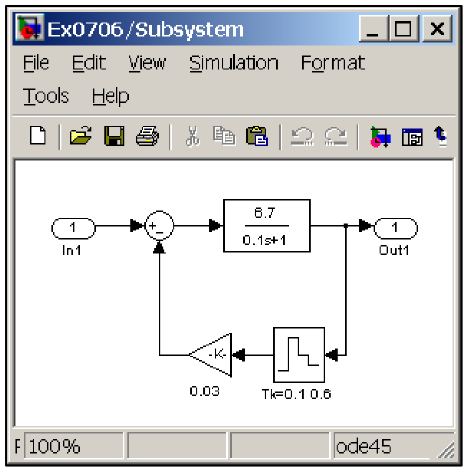


图 7.38 子系统模块窗口

然后将系统模型修改为如图 7.39 所示的系统。

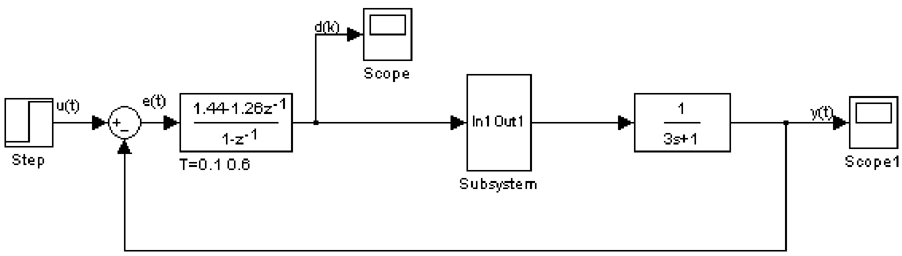
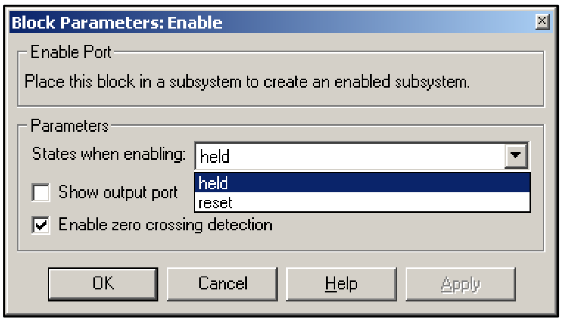


图 7.39 包含子系统的模型

创建的子系统可以打开和修改，但不能再解除子系统设置。

7.6.2 条件执行子系统

1. 使能子系统(Enabled Subsystem)



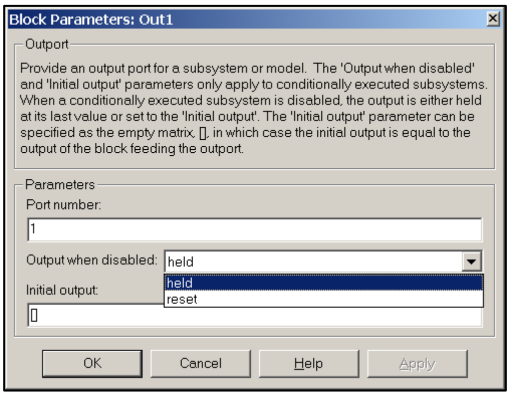
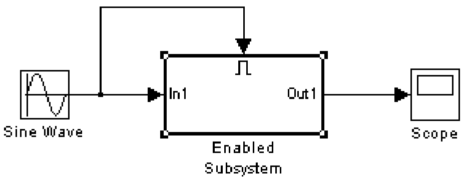


图 7.41 (a) “Enable”模块参数设置　　　　(b) “Out1”模块参数设置

【例 7.8】建立一个用使能子系统控制正弦信号为半波整流信号的模型。

模型由正弦信号“Sine wave”为输入信号源，示波器“Scope”为接收模块，使能子系统“Enabled Subsystem”为控制模块，



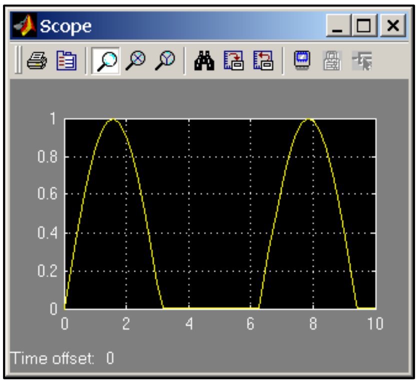


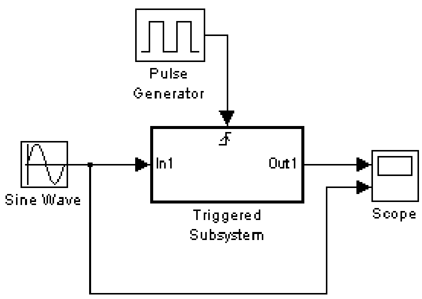
图 7.42 (a) 使能子系统模型　　　　(b) 示波器显示

连接模块，将“Sine wave”模块的输出作为“Enabled Subsystem”的控制信号，模型如图 7.42(a)所示。

开始仿真，由于“Enabled Subsystem”的控制为正弦信号，大于零时执行输出，小于零时就停止，则示波器显示为半波整流信号，示波器的显示如图 7.42(b)所示。

2. 触发子系统(Triggered Subsystem)

【例 7.9】建立一个用触发子系统控制正弦信号输出阶梯波形的模型。



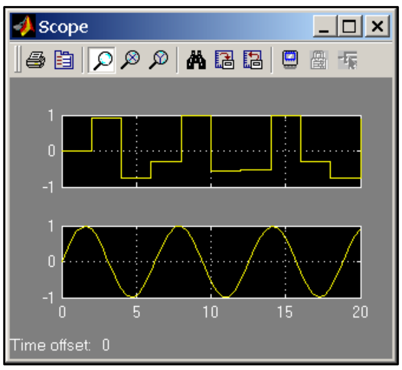


图 7.43 (a) 触发子系统模型　　　　(b) 示波器显示

模型由正弦信号“Sine wave”为输入信号源，示波器“Scope”为接收模块，触发子系统“Triggered Subsystem”为控制模块，选择“Sources”模块库中的“Pulse Generator”模块为控制信号。

连接模块，将“Pulse Generator”模块的输出作为“Triggered Subsystem”的控制信号，模型如图 7.43(a)所示。

开始仿真，由于“Triggered Subsystem”的控制为“Pulse Generator”模块的输出，示波器输出如图 7.43(b)所示。

3. 使能触发子系统(Enabled and Triggered Subsystem)

使能触发子系统就是触发子系统和使能子系统的组合，含有触发信号和使能信号两个控制信号输入端，触发事件发生后，Simulink 检查使能信号是否大于 0，大于 0 就开始执行。

7.6.3 子系统的封装

1. 封装子系统的步骤

(1) 选中子系统双击打开，给需要进行赋值的参数指定一个变量名；

(2) 选择菜单“Edit”→“Mask subsystem”，出现封装对话框；

(3) 在封装对话框中的设置参数，主要有“Icon”、“Parameters”、“Initialization”和“Documentation”四个选项卡。

2. Icon 选项卡

Icon 选项卡用于设定封装模块的名字和外观，如图 7.44 所示。

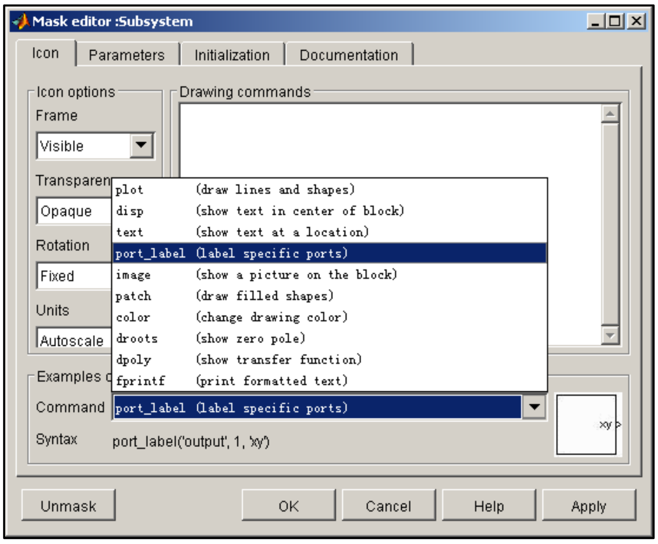


图 7.44 Icon 参数设置

(1) Drawing commands 栏

用来建立用户化的图标，可以在图标中显示文本、图像、图形或传递函数等。在 Drawing commands 栏中的命令如上图中“Examples of drawing commands”的下拉列表所示，包括 plot、disp、text、port_label、image、patch、color、droots、dploy 和 fprintf。

(2) Icon Options 栏

用于设置封装模块的外观。

3. Parameters 选项卡

Parameters 选项卡用于输入变量名称和相应的提示，如图 7.45 所示。

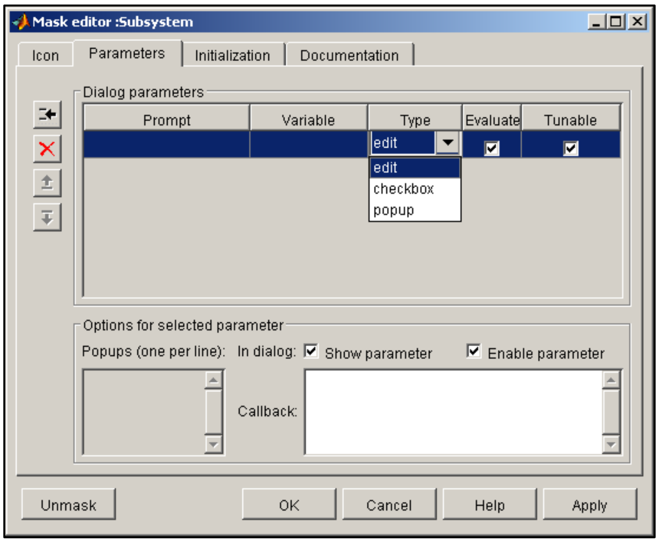
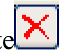


图 7.45 Parameters 参数设置

(1) Add、Delete、Move up和 Move down按钮

用于添加、删除、上移和下移输入变量。

(2) Dialog Parameters

- Prompt：输入变量的含义，其内容会显示在输入提示中。
- Variable：输入变量的名称。
- type：给用户提供设计编辑区的选择。“Edit”提供一个编辑框；“Checkbox”提供一个复选框；“Popup”提供一个弹出式菜单。
- Evaluate：用于配合“type”的不同选项提供不同的变量值，有两个选项“Evaluate”和“Literal”，其含义如表 7.6 所示。

表 7.6 Assignment 选项的不同含义

Evaluate / type	on	off
Edit	输入的文字是程序执行时所用的变量值	将输入的内容作为字符串
Checkbox	输出 1 和 0	输出为 on 或 off
Popup	将选择的序号作为数值，第一项则为 1	将选择的内容当作字符串

(3) Options for selected parameter

- Pops：当“type”选择“Popup”时，用于输入下拉菜单项。
- Callback：用于输入回调函数。

4. Initialization 选项卡

Initialization 选项卡用于初始化封装子系统。

5. Documentation 选项卡

Documentation 选项卡用于编写与该封装模块对应的 Help 和说明文字，分别有“Mask type”、“Mask Description”和“Mask help”栏。

(1) Mask type 栏

用于设置模块显示的封装类型。

(2) Mask Description 栏

用于输入描述文本。

(3) Mask help 栏

用于输入帮助文本。

6. 按钮

设置参数设置对话框中的“Apply”按钮用于将修改的设置应用于封装模块；“Unmask”按钮用于将封装撤销，则双击该模块就不会出现定制的对话框。

【例 7.10】创建一个二阶系统，并将子系统进行封装。

创建一个二阶系统，将其闭环系统构成子系统，并封装将阻尼系数 zeta 和无阻尼频率 wn 作为输入参数。

(1) 创建模型，并将系统的阻尼系数用变量 zeta 表示，无阻尼频率用变量 wn 表示，如图 7.46 所示。

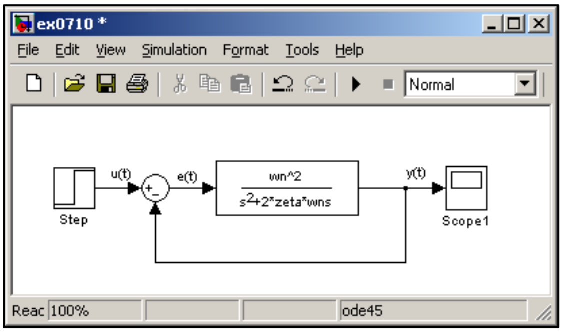


图 7.46 二阶系统模型

(2) 用虚线框框住反馈环，选择菜单“Edit”→“Create Subsystem”，则产生子系统，如图 7.47 所示。

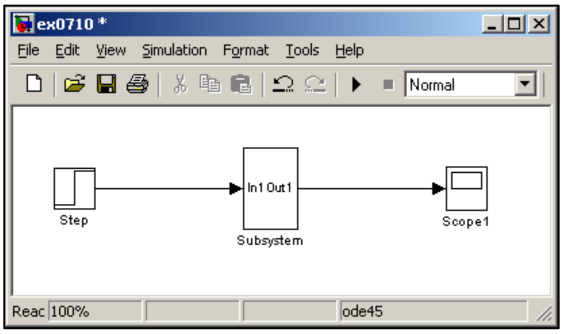


图 7.47 子系统模型

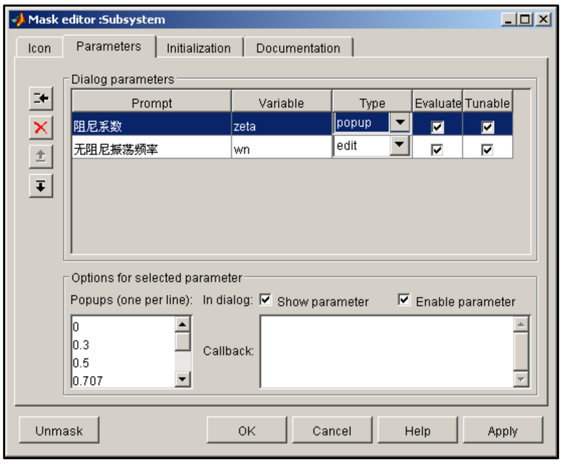
(3) 封装子系统，选择菜单“Edit”→“Mask subsystem”，出现封装对话框，将 zeta 和 wn 作为输入参数。

在 Icon 选项卡中设置的“Drawing commands”栏中写文字并画曲线，命令如下：

```
disp('二阶系统')
plot([0 1 2 3 10],-exp(-[0 1 2 3 10]))
```

在 Parameters 选项卡中，单击“Add”按钮添加两个输入参数，设置“Prompt”分别为“阻尼系数”和“无阻尼振荡频率”，并设置“type”栏分别为“Popup”和“edit”，对应的“Variable”为“zeta”和“wn” ，设置“Popups”为“0 0.3 0.5 0.707 1 2”，如图 7.48(a)所示。

在 Initialization 选项卡初始化输入参数，如图 7.48(b)所示。



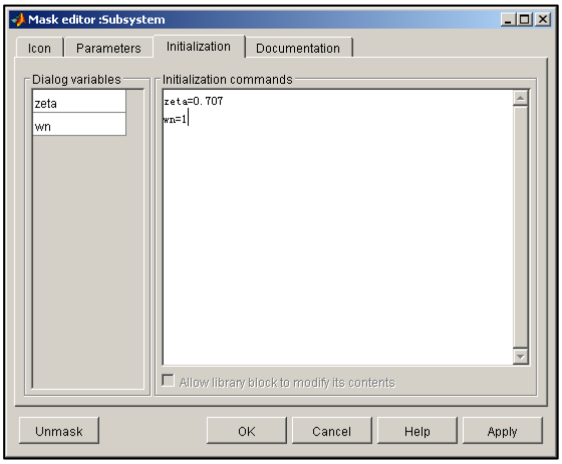


图 7.48 (a) Parameters 选项卡 **(b) Initialization 选项卡**

在 Documentation 选项卡中输入提示和帮助信息，如图 7.48(c)所示。

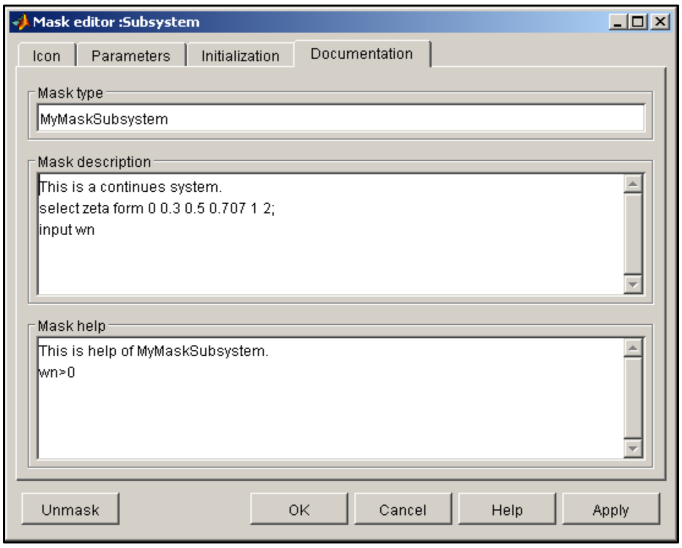
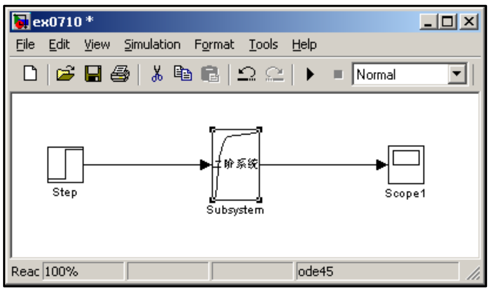


图 7.48(c) Documentation 选项卡

单击“OK”按钮，完成参数设置，然后双击该封装子系统，则出现如图 7.49(a)所示的封装子系统，双击该子系统出现图 7.49(b)所示的输入参数对话框，在对话框中输入“阻尼系数”zeta 和“无阻尼振荡频率”wn 的值，再不需要为子系统中的每个模块分别打开参数设置对话框了。



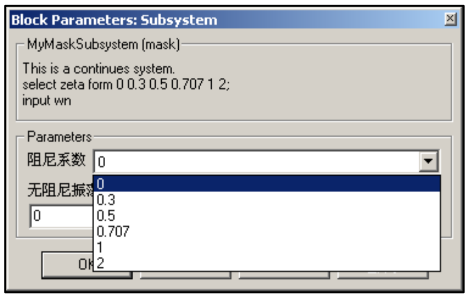


图 7.49 (a) 封装子系统外观　　　　(b) 封装子系统参数输入对话框

7.7 用 MATLAB 命令创建和运行 Simulink 模型

7.7.1 用 MATLAB 命令创建 Simulink 模型

1. Simulink 模型与文件

(1) 创建新模型

new_system 命令用来在 MATLAB 的工作空间创建一个空白的 Simulink 模型。

语法：

new_system(‘newmodel’,option)　　　　　**%创建新模型**

说明：‘newmodel’为模型名；option 选项可以是’Library’和’Model’两种，也可以省略，默认为’Model’。

(2) 打开模型

open_system 命令用来打开逻辑模型，在 Simulink 模型窗口显示该模型。

语法：

open_system(‘model’)　　　　　**%打开模型**

说明：‘model’为模型名。

(3) 保存模型

save_system 命令用来保存模型为模型文件，扩展名为.mdl。

语法：

save_system(‘model’,文件名)　　　　　**%保存模型**

说明：‘model’为模型名可省略，如果不给出模型名，则自动保存当前的模型；文件名指保存的文件名，是字符串，也可省略，如果不省略则保存为新文件。

【例 7.11】用 MATLAB 命令创建新模型。

```
new_system('Ex0711model')              %创建逻辑模型
open_system('Ex0711model')                  %打开模型
save_system('Ex0711model','Ex0711')         %保存模型文件
```

2. 添加模块和信号线

(1) 添加模块

使用 add_block 命令在打开的模型窗口中添加新模块。

语法：

add_block(’源模块名’，’目标模块名’,’属性名 1’,属性值 1，’属性名 2’,属性值 2,…)

说明：’源模块名’为一个已知的库模块名，或在其它模型窗口中定义的模块名，Simulink自带的模块为内在模块，例如正弦信号模块为’built-in/Sine Wave’；’目标模块名’为在模型窗口中使用的模块名。

(2) 添加信号线

模块需要用信号线连接起来，添加信号线使用 add_line 命令。

语法：

add_line(’模块名’,’起始模块名/输出端口号’, ’终止模块名/输入端口号’)

add_line(’模块名’,m)

说明：’模块名’为在模型窗口中的模块名；m 为有两列元素的矩阵，每列给出一个转折点坐标。

【**例 7.11 续**】用 MATLAB 命令添加四个模块连接成一个二阶系统模型。

```
add_block('built-in/Step','Ex0711/Step','position',[20,100,40,120])      %添加阶跃信号模块
add_block('built-in/Sum','Ex0711/Sum','position',[60,100,80,120])       %添加 Sum 模块
add_block('built-in/Transfer Fcn','Ex0711/Fcn1','position',[120,90,200,130])
%添加传递函数模块
add_block('built-in/Scope','Ex0711/Scope','position',[240,100,260,120])   %添加示波器模块
add_line('Ex0711','Step/1','Sum/1')                                       %添加连线
add_line('Ex0711','Sum/1','Fcn1/1')
add_line('Ex0711','Fcn1/1','Scope/1')
add_line('Ex0711','Fcn1/1','Sum/2')
```

程序分析：'position'为位置属性，模块名为'Ex0711'。

则出现如图 7.50 所示的模型。

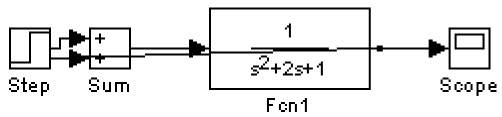


图 7.50 二阶系统模型

3. 设置模型和模块属性

(1) 模型属性的获得

【**例 7.11 续**】获得模型属性和各属性的含义。

```
f1=simget('Ex0711')
f1 =
            AbsTol: 'auto'             %绝对允许误差限
             Debug: 'off'              %是否允许跟踪调试
        Decimation: 1                  %输出位数，每个 1 点输出 1 次
      DstWorkspace: 'current'          %输出量工作空间
    FinalStateName: ''                 %状态变量名
         FixedStep: 'auto'             %定步长
      InitialState: []                 %初始状态向量
       InitialStep: 'auto'             %初始步长
          MaxOrder: 5                  %最高算法阶次
        SaveFormat: 'Array'            %变量类型
     MaxDataPoints: 1000               %最大返回点数
           MaxStep: 'auto'             %最大步长
```

```
            MinStep: []                 %最小步长
       OutputPoints: 'all'              %输出点
    OutputVariables: 'ty'               %输出变量
             Refine: 1                  %插值点
             RelTol: 0.0010             %相对误差
             Solver: 'ode45'            %仿真算法
       SrcWorkspace: 'base'             %输入量工作空间
              Trace: ''                 %是否逐步显示
          ZeroCross: 'on'               检测过零点
```

(2) 设置模型属性

(3) 设置模块和信号线属性

【例 7.11 续】设置各模块的属性，建立一个与【例 7.3】模型参数相同的二阶系统模型。

```
set_param('Ex0711','StopTime','15')                    %设置采样停止时间
set_param('Ex0711/Step','time','0')                    %设置阶跃信号上升时间
set_param('Ex0711/Sum','Inputs','+-')                  %设置 Sum 模块信号的符号
set_param('Ex0711/Fcn1','Denominator','[1 0.6 0]')     %设置传递函数分母
```

则系统模型框图如图 7.51 所示。

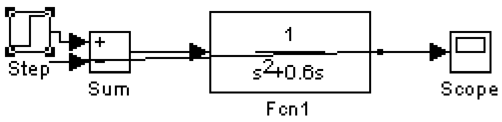


图 7.51 二阶系统模型

4. 删除模块和信号线

(1) 删除模块

例如删除示波器模块则使用：

```
delete_block('Ex0711/Scope')
```

(2) 删除信号线

7.7.2 用 MATLAB 命令运行 Simulink 模块

使用 sim 命令来完成，在命令窗口就可以方便地对模型分析和仿真。

语法：

[t,x,y]=sim('model',timespan,options,ut)　　**%利用输入参数进行仿真，输出矩阵**

[t,x,y1,y2,…]=sim('model',timespan,options,ut)**%利用输入参数进行仿真，逐个输出**

说明：'model'为模型名；timespan 是仿真时间区间，可以是[t0,tf]表示起始时间和终止时间，也可以是[]，利用模型对话框设置时间，如果是标量则指终止仿真时间；options 参数为模型控制参数；ut 为外部输入向量。t 为时间列向量；x 为状态变量构成的矩阵；y 为输出信号构成的矩阵，每列对应一路输出信号。timespan 、options 和 ut 参数都可省略。

【例 7.11 续】运行二阶系统的阶跃响应，如图 7.50 所示。

```
[t,x,y]=sim('Ex0711',[0,15]);
plot(t,x(:,2))
```

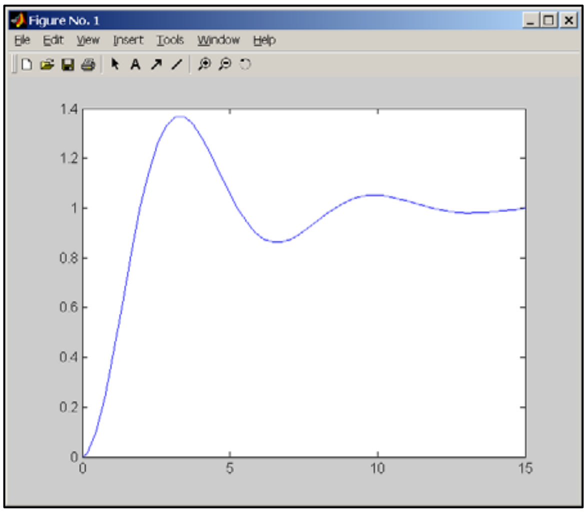


图 7.50 二阶系统时间响应

7.8 以 Simulink 为基础的模块工具箱简介

在打开 Simulink 时出现的界面中，如前面的图 7.1 所示左侧的模块库和工具箱(Block and Toolboxes)栏中列出了各领域开发的仿真环节库。

主要的仿真环节库有：

- 控制系统工具箱(Control System Toolbox)
- 通信模块工具箱(Communications Blockset)
- 数字信号处理模块工具箱(DSP Blockset)
- 非线性控制模块工具箱(NCD Blockset)
- 定点处理模块工具箱(Fixed-Point Blockset)
- 状态流(StateFlow)
- 系统辨识模块工具箱(System ID Blocks)
- 神经网络模块工具箱(Neural Network Blockset)
- 模糊逻辑工具箱(Fuzzy Logic Toolbox)

第 8 章 MATLAB 高级应用

8.1 MATLAB 应用接口

8.1.1 MEX 文件

MEX 文件具有以下几个方面的应用：

(1) 对于已存在的 C 或 FORTRAN 子程序，可以通过 MEX 文件在 MATLAB 环境中直接调用，而不必重新编写 M 文件。

(2) 由于 MATLAB 是解释性语言，运行如 for 等循环体时，会出现速度十分缓慢的现象，为了能提高速度，往往要使用 MEX 程序。

(3) 对于 A/D、D/A 卡，或其它 PC 硬件，可以直接用 MEX 文件进行访问。

(4) 利用 MEX 文件，可以使用如 Windows 用户图形界面等资源。

1. MEX 文件系统设置

下面采用 Microsoft Visual C/C++ 6.0 编译器，在命令窗口使用“mex -setup”命令，对 MEX 文件编译器进行配置：

```
>> mex -setup
```

则会出现如下提示：

```
Please choose your compiler for building external interface (MEX) files:
Would you like mex to locate installed compilers [y]/n? y
```

让用户选择是否同意 mex 命令自动定位已经安装的编译器，输入“y”后，回车出现如下提示：

```
Select a compiler:
[1] Digital Visual Fortran version 6.0 in C:\Program Files\Microsoft Visual Studio
[2] Lcc C version 2.4 in D:\MATLAB6P1\sys\lcc
[3] Microsoft Visual C/C++ version 6.0 in D:\Program Files\Microsoft Visual Studio

[0] None
```

提供用户选择编译器作为默认的 MEX 文件编译器，通过键盘输入“3”后，回车出现如下提示：

```
 Please verify your choices:
 Compiler: Microsoft Visual C/C++ 6.0
Location: D:\Program Files\Microsoft Visual Studio
Are these correct?([y]/n):
```

确认所选择的编译器，通过键盘输入“y”后，回车出现如下提示：

```
The default options file:
"D:\Documents and Settings\ZJDCY\Application Data\MathWorks\MATLAB\R12\mexopts.bat"
is being updated from D:\MATLAB6P1\BIN\WIN32\mexopts\msvc60opts.bat...
Installing the MATLAB Visual Studio add-in ...
  Updated D:\Program Files\Microsoft Visual Studio\common\msdev98\template\MATLABWizard.awx
    from D:\MATLAB6P1\BIN\WIN32\MATLABWizard.awx
  Updated D:\Program Files\Microsoft Visual Studio\common\msdev98\template\MATLABWizard.hlp
    from D:\MATLAB6P1\BIN\WIN32\MATLABWizard.hlp
  Updated D:\Program Files\Microsoft Visual Studio\common\msdev98\addins\MATLABAddin.dll
    from D:\MATLAB6P1\BIN\WIN32\MATLABAddin.dll
  Merged D:\MATLAB6P1\BIN\WIN32\usertype.dat
    with D:\Program Files\Microsoft Visual Studio\common\msdev98\bin\usertype.dat
```

则表示编译器成功配置。

2. 测试配置

在 MATLAB 环境的当前目录浏览器窗口中，将 MATLAB 的 extern\examples\mex 目录设置为当前目录，然后在命令窗口中输入：

```
mex yprime.c
```

则在 extern\examples\mex 目录下，就生成了“yprime.dll”文件，查看该 yprime.dll 文件的信息，在命令窗口中输入：

```
>> which yprime
D:\MATLAB6p1\extern\examples\mex\yprime.dll
```

并调用 yprime.dll 文件：

```
>> yprime(1,1:2:7)
ans =
    3.0000    14.9925    7.0000    -1.0377
```

3. C 语言 MEX 文件的构成

程序的构成主要由入口子程序和计算功能子程序两部分组成。

(1) 入口子程序

第一部分入口子程序的作用是在 MATLAB 系统与被调用的外部子程序之间建立通信联系。入口子程序必须是 mexFunction，其构成形式为：

```
void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray*prhs[] )
{
    /*用来完成 MATLAB 与计算子程序之间通信任务的代码*/
}
```

mexFunction 函数的参数含义为：

- nrhs：为输入参数的数目。
- prhs：为输入参数数组，是指针数组。
- nlhs：为输出参数的数目。
- plhs：为输出参数数组，是指针数组。

(2) 计算功能子程序

第二部分为计算功能子程序，包含所有的实际需要完成的功能源代码，可以是用户以前所编写的算法和程序，以函数的形式存在。

4. C 语言 MEX 文件的建立

(1) 常用 MATLAB API 函数

在 MEX 文件中，常用 MATLAB API 函数有：

- mxGetM：获得矩阵的行数。
- mxGetN：获得矩阵的列数。
- mxGetPr：获得矩阵的实数部分的数据指针。
- mexErrMsgTxt：输出错误信息，并返回到 MATLAB 命令提示符下。
- mxCreateString：创建一个字符串矩阵。
- mxCreateDoubleMatrix：创建一个二维未赋值的双精度浮点类型的矩阵。
- mxIsDouble：判断矩阵是否为双精度类型。

(2) C 语言 MEX 文件的创建步骤

通过创建计算一个数平方倒数值的程序，来介绍一个 MEX 文件创建的过程。

【例 8.1】在 Microsoft Visual C++6.0 环境下创建一个 MEX 文件，实现计算一个数平方的倒数值。

- 在 Microsoft Visual C++6.0 环境创建一个“C++ Source File”文件，文件名为“Ex0801.cpp”。

■ 编写该文件的程序代码如下：

```
#include "mex.h"
#include "math.h"

/*计算功能子程序 Ex0801，计算平方的倒数*/
void Ex0801(double y[],double x[])
{
        y[0]=1/(x[0]*x[0]);
        return;
}

/*入口子程序 mexFunction*/
void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray *prhs[])
{
        double *x,*y;
        unsigned int m,n;
        /*检查输入变量的个数是否正确*/
        if(nrhs!=1)
        {
                mexErrMsgTxt("Only one input argument allowed.");
        }
        /*检查输出变量的个数是否正确*/
        else if(nlhs!=1){
                mexErrMsgTxt("Only one output argument allowed.");
        }
       m = mxGetM(prhs[0]);
       n = mxGetN(prhs[0]);
        /*检查输入变量必须是非复数单个双精度数*/
        if (!mxIsDouble(prhs[0]) || mxIsSparse(prhs[0]) ||mxIsComplex(prhs[0])||
                ! (m==1&&n==1))
        {
                mexErrMsgTxt("Input argument must be a scalar.");
        }
        /*创建矩阵变量为输出变量*/
        plhs[0]=mxCreateDoubleMatrix(m,n,mxREAL);
        y = mxGetPr(plhs[0]);
        x = mxGetPr(prhs[0]);
        /*调用计算功能子程序*/
        Ex0801(y,x);
}
```

■ 在 MATLAB 命令窗口的当前目录浏览器中将当前目录设置为用户的目录，输入命令创建 MEX 文件并运行：

```
  mex Ex0801.cpp
>> y=Ex0801(2)
y =
       0.2500
```

扩展名为 cpp 是 C++文件，在其同一文件夹中生成 Ex0801.dll 文件，运行 Ex0801.dll 文件的运算结果正确。

■ 为该文件添加帮助 M 文件

为了在 MATLAB 中可以方便地查看该 Ex0801.dll 文件的帮助信息，可以建立一个 Ex0801.M 文件，并输入以下内容：

```
%function y=Ex0801(x)
%计算一个数平方的倒数值
% y=1/(x^2)
% copyright @ 2003-10-10
```

则在 MATLAB 命令窗口中，用 help 命令只能看到 M 文件来查看帮助信息：

```
>> help Ex0801
 function y=Ex0801(x)
  计算一个数平方的倒数值
  y=1/(x^2)
  copyright @ 2003-10-10
```

8.1.2 使用 MATLAB 编译器生成 MEX 和 EXE 文件

1. MEX 文件与 EXE 文件的区别

MEX 文件只能在 MATLAB 环境中运行；而 EXE 文件是可以独立与 MATLAB 环境运行的。

2. 配置编译器的准备

如果要创建 MEX 文件，如图 8.1 所示。则只需勾选“MATLAB Compiler”；如果要创建 EXE 文件，则需要勾选如图中的三个选项。

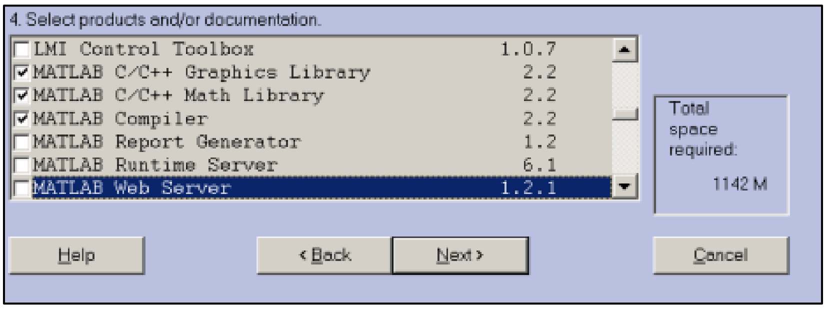


图 8.1 安装 MATLAB 对话框

3. 由 M 文件创建 C MEX 文件

利用 MATLAB 编译器来由 M 文件创建 MEX 文件，使用“mcc –x”命令，。

【例 8.2】在 MATLAB 环境中创建一个 M 函数文件，用来绘制正弦信号波形。

在 MATLAB 的编辑/调试窗口中创建一个 Ex0802.m 函数文件，编写程序代码如下：

```
function Ex0802
% Ex0802  绘制正弦信号波形
x=0:0.1:2*pi;
y=sin(x);
plot(x,y)
```

保存文件，在 MATLAB 命令窗口中由该 M 文件创建 MEX 文件：

```
>> mcc -x Ex0802          %创建 Ex0802.dll 文件
>> Ex0802                 %运行 Ex0802.dll 文件
```

运行该 Ex0802.dll 文件，绘制出正弦信号波形图。在该目录下，可以看到在创建 dll 文件的同时，还创建了 Ex0802.c、Ex0802.h 和 Ex0802_mex.c 文件。

注意：如果上述的 Ex0802.m 文件删除第一行，变为 M 脚本文件，则用 mcc 命令直接编译会出错，并出现如下提示：

```
>> mcc -x Ex0802
??? Error: Cannot create MEX File "Ex0802" from MEX File "C:\My Documents\CAOYi\BookMATLAB\20031008\exe\Ex0802.dll".
Error in ==> D:\MATLAB6p1\toolbox\compiler\mcc.dll
```

4. 创建独立的外部程序 EXE 文件

【例 8.3】第一次使用 mbuild 命令由 C 文件创建 EXE 文件。

由“matlab\extern\examples\cmath”目录下的现成文件“ex1.c”，创建独立的 EXE 文件。

先将“matlab\extern\examples\cmath”目录设置为当前目录，然后输入 mbuild 命令，由于系统中安装了 LCC 和 Microsoft Visual C/C++两个编译器，则 MATLAB 会提示用户选择默认的编译器：

```
>> mbuild ex1.c
Please choose your compiler for building standalone MATLAB applications:
Would you like mbuild to locate installed compilers [y]/n? y
```

选择一个默认的编译器

```
Select a compiler:
[1] Lcc C version 2.4 in D:\MATLAB6P1\sys\lcc
[2] Microsoft Visual C/C++ version 6.0 in D:\Program Files\Microsoft Visual Studio
[0] None
Compiler: 2
……
```

后面就会出现提示和确认的信息。

如果要改变默认的编译器，则需要使用“mbuild -setup”命令重新选择。

(1) mbuild 命令由 C 文件生成 EXE 文件

【例 8.3 续】用 mbuild 命令由 C 文件编译生成 EXE 文件。

使用 mbuild 命令可以由“ex1.c”文件创建独立的可执行文件：

```
>> mbuild ex1.c
```

则产生了“ex1.exe”文件，可以脱离 MATLAB 独立运行，同时在同一目录下还产生了“bin”文件夹。

(2) mcc 命令由 M 文件生成 EXE 文件

【例 8.4】用 mcc 命令由【例 8.2】中的 M 文件编译生成 EXE 文件。

```
>> mcc -B sglcpp Ex0802.m          %创建需要图形库的 C++独立应用程序
```

如果不希望 MCC 自动调用 mbuild，可以添加使用“-c”选项，如命令“mcc -mc”和“mcc -pc”。

8.2 低级文件的输入输出

8.2.1 打开和关闭文件

1. 打开文件

语法：

fid = fopen(filename,permission) %以指定格式打开文件

[fid,message] = fopen(filename,permission) %返回打开文件的信息

说明：fid 为返回的文件指针(File Identifier)，通常是一个非负的整数，如果返回-1，则表示无法打开文件；message 用来显示打开文件的信息，如果无法打开，则显示错误信息；filename 为文件名，是字符串，如果文件不在 MATLAB 的搜索路径中，则需要指定文件路径；permission 为指定文件的打开模式，有以下几种模式：

- 'r'：只读文件
- 'r+'：读写文件
- 'w'：删除已存在文件内容或建立新文件，并只写文件
- 'w+'：删除已存在文件内容或建立新文件，并读写文件
- 'a'：以只写方式建立并打开一个新文件或打开一个已存在的文件，只能在文件末尾添加内容
- 'a+'：以读写方式建立并打开一个新文件或打开一个已存在的文件，在文件末尾添加内容

fopen 的 permission 参数在打开文件时还可标明文件格式，如果打开文本格式文件，在后添加字母“t”，如果打开二进制格式文件，则在 permission 参数后添加字母“b”，如'wb'、'rb+'等。

2. 关闭文件

打开文件进行读写操作后，应立即关闭文件，删除文件指针。

语法：

status=fclose(fid) %关闭文件指针所指的文件

status=fclose('all') %关闭所有打开的文件

说明：status 为关闭文件指针所指文件的状态，如果成功则返回 0，如果失败则返回-1；fid 为所打开的文件指针。

【例 8.5】打开和关闭一个文本文件。

文本文件“Ex0805.txt”，在 MATLAB 环境中显示的文件内容如下：

```
type Ex0805.txt

a 1 2 3
b 4 5 6
```

使用 fopen 和 fclose 命令打开和关闭文件：

```
[fid,message]=fopen('Ex0805.txt','w+')       %打开文件读写

fid =
    3
message =
```

```
    ''
 if fid==-1
disp('无法打开该文件')
else
disp('成功打开该文件')
end

成功打开该文件
status=fclose(fid)                       %关闭文件

status =
    0
```

8.2.2 读写格式化文件

1. fscanf 命令

fscanf 命令为读格式化文件数据。

语法：

[a,count]=fscanf(fid,format,size) %读取格式化数据

说明：fid 为文件指针，所指为需要读取的格式化文件；format 指定读取数据格式，指定的格式必须和文件中的数据格式相同，否则读取的数据可能会出现错误，以“%”开头，有%c、%d、%e、%f、%g、%i、%o、%s、%u、%x 等(与 C 语言相同)；count 为成功读取的数据元素个数，可省略；a 返回读取的数据；size 为需要读取的数据个数，如果省略，则读到文件末尾，size 的取值可以有：

- n：读 n 个数据到一个列向量
- inf：读到文件末尾，数据放到一个列向量
- [m,n]：读出的数据个数为 m×n，数据放到矩阵中，读出的数据按列的顺序填充矩阵，不够的数据用 0 填补。

【例 8.5 续】读取 Ex0805.txt 文件的前四个字符。

```
 fid=fopen('Ex0805.txt')

fid =
    3

 a1=fscanf(fid,'%s',4)                   %以字符串格式读取四个数据

a1 =
a123
 fclose(fid)

ans =
    0
```

2. fprintf 命令

fprintf 命令为写格式化数据。

语法：

count=fprintf(fid,format,a,…) %写入格式化数据

说明：fid 为文件指针，所指为二进制文件；a 为矩阵数据，将 a 写到 fid 指向的文件；format 为写入的格式，除了包含 fscanf 命令的数据格式之外，还有%E、%G、%X，并具有对齐格式-(左对齐)、+(右对齐)、0(补齐位数)，还有转义字符；count 为成功写入数据的个数。

【例 8.6】使用 Ex0805.txt 文件进行读取和写入数据。

```
a='%This is a example.';
fid=fopen('Ex0805.txt','a+')        %打开 Ex0803.txt 文件在末尾添加

fid =
     3
 fprintf(fid,'%s',a)                %写入 a 到文件末尾

ans =
    19
 fclose(fid)                        %关闭文件

ans =
     0
 fid=fopen('Ex0805.txt','r')            %打开 Ex0803.txt 文件只读

fid =
     3
 fscanf(fid,'%s')                   %读取文件所有内容

ans =
a123b456%Thisisaexample.%Thisisaexample.%Thisisaexample.%Thisisae
xample.
 fclose(fid)

ans =
     0
```

程序分析：在向文件中写入数据后，先关闭文件，然后再打开文件则从文件开头读取数据，如果写完数据后直接读取数据，则实际读取数据的位置将从写入的最后一个数据之后开始。

3. fgetl 和 fgets 命令

fgetl 和 fgets 命令都是用来读取文件的下一行，两者的差别是 fgetl 会舍去换行符，而 fgets 则保留换行符。

语法：

tline=fgetl(fid) %读取文件的下一行，不包括换行符
tline=fgets(fid) %读取文件的下一行，包括换行符
tline=fgets(fid,nchar) %限制读取文件字符个数

说明：fid 为文件指针；tline 为以字符串形式的返回值，如果到文件末尾则返回-1；nchar 为最多返回的字符个数。

【例 8.6 续】以行的形式读取 Ex0805.txt 文件。

```
fid=fopen('Ex0805.txt','r') ;          %打开 Ex0803.txt 文件只读
fgetl(fid)                             %读取第一行数据

ans =
a 1 2 3
fgets(fid)                             %读取第二行数据

ans =
b 4 5 6
fgets(fid,10)                          %读取第三行数据，限制 10 个字符

ans =
%This is a
fgets(fid,10)

ans =
 example.%
```

8.2.3 读写二进制数据

fread 命令为读二进制数据。

语法：

[a,count]=fread(fid,size,precision,skip)　　　　%读取二进制数据

说明：fid 为文件指针；size 与 fscanf 命令含义相同；precision 为一个字符串，用来指定读取数据的精度，即数据类型，有'uchar'、'schar'、'int8'、'int16'、'int32'、'int64'、'unit8'、'unit16'、'unit32'、'unit64'、'single'、'float32'、'double'、'float64'等，可省略；a 为矩阵数据；count 为成功读取的数据元素个数，可省略；skip 为每读取一个数据后跳过的字节数，可省略。

2. 写数据

fwrite 命令为写二进制数据。

语法：

count=fwrite(fid,a,precision,skip)　　　　%写二进制数据

说明：fid 为文件指针；a 为矩阵数据；precision 和 skip 参数含义与 fread 命令相同；count 为成功写入数据的个数。

【例 8.7】写入数据到 MAT 文件中，并读取数据。

```
x1=1:10;
[fid,message]=fopen('Ex0805.mat','a')          %打开文件添加数据

fid =
     4
```

```
message =
     ''

 count1=fwrite(fid,x1)                        %写入数据

count1 =
    10

 x2=11:15;
 count2=fwrite(fid,x2)                        %添加数据

count2 =
     5
 status=fclose(fid);

 fid=fopen('Ex0805.mat','r');                 %打开文件只读
 a1=fread(fid,[2,5])                          %读取数据

a1 =
     1     3     5     7     9
     2     4     6     8    10
 a2=fread(fid,[1,5])

a2 =
     1     2     3     4     5
 fclose(fid);
```

8.2.4 文件定位

1. fseek 命令

fseek 命令用来移动文件位置指针。

语法：

status=fseek(fid,offset,origin)　　　　　　%移动文件位置指针

说明：fid 为文件指针；offset 指定移动的字节数，如果 offset >0，则向后移动，否则向前移动，等于 0 则不移动；status 为返回值，如果移动成功则返回 0，否则返回-1；origin 指定移动位置指针的参考起点：

- 'bof' 或-1：文件的开头
- 'cof' 或 0：文件的当前位置
- 'eof' 或 1：文件的末尾。

2. ftell 命令

ftell 命令是用来获取文件位置指针的当前位置。

语法：

pos=ftell(fid)　　　　　　%获取当前指针位置

说明：pos 指字节数，当前位置指针指在此字节数之后。

3. frewind 命令

frewind 命令用来将文件位置指针移到文件的开头。

语法：

frewind(fid)

4. feof 命令

feof 命令用来测试位置指针是否在文件结束位置，如果是则返回 1，否则返回 0。

语法：

feof(fid)

【例 8.8】创建两个 mat 文件，在 Ex0808_1.mat 文件中写入 1～10 的数据，并进行求和，在 Ex0808_2.mat 文件中写入 1、2、3 三个数据，将第二个数据与前面所求的和进行相乘运算。

程序保存在 Ex0808.m 文件中，程序代码如下：

```
% Ex0808    文件读取和定位
x=1:10;
s=0;
fid1=fopen('Ex0808_1.mat','w+')             %打开文件读写数据
fwrite(fid1,x);                                     %写入数据
frewind(fid1);                                      %指针移到文件开头
while feof(fid1)==0                             %判断是否到文件末尾
       a1=fread(fid1,1)                         %读取数据
       if isempty(a1)==0                        %判断是否为空值
                 s=a1+s                                 %求和
        end
end
fclose(fid1);
y=[1 2 3];
fid2=fopen('Ex0808_2.mat','w+')             %打开文件读写数据
fwrite(fid2,y)                                  %写入数据
fseek(fid2,-2,'eof')                            %指针移动到第二个数据
a2=fread(fid2,1)                                %读取数据
s=s*a2
fclose(fid2);
```

运行结果得出：

s =

    110

程序说明：

- 使用文件位置控制就可以不用反复打开和关闭文件，而直接从文件中读写数据。
- 使用 while 循环结构，从文件中读取数据，直到文件末尾。
- 当文件位置指针移动到文件最后时，取出的数据为空值，但 feof 函数返回 0，因此用 isempty 函数判断是否为空值来判断是否到文件最后，文件指针再向下移则到文件末尾，feof 函数返回 1。
- “fseek(fid2,-2,'eof')”语句是将文件位置指针从末尾向前 2 个数据。

8.3 图形文件的转储

MATLAB 生成的图形文件为.fig 文件格式，有时需要将图形文件转储为多种常用的标准格式，如 tif、bmp、jpg 等。选择菜单“File”→“Export”命令，则在出现的“Export”对话框中，在“保存类型”栏选择需要转储的图形文件类型，如图 8.2 所示。

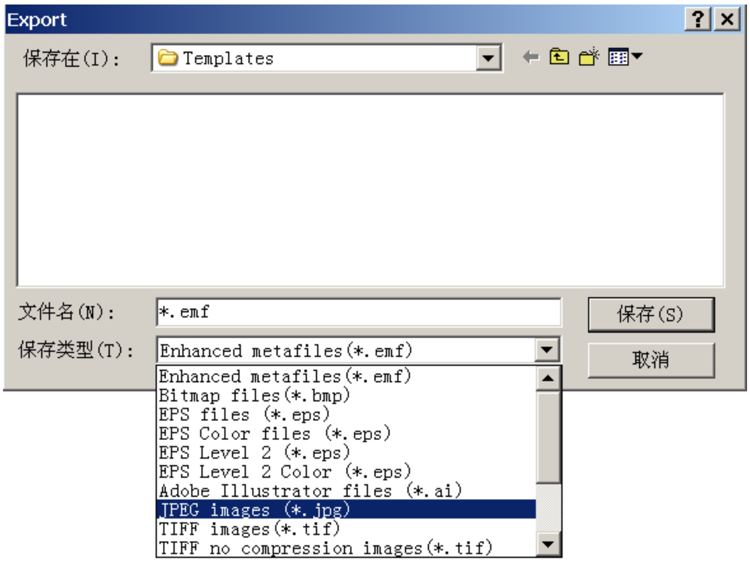


图 8.2 “Export”对话框

8.4 Notebook

8.4.1 Notebook 的安装

MATLAB 6.5 的 Notebook 安装步骤如下：

(1) 启动 MATLAB，出现 MATLAB 命令窗口。

(2) 在命令窗口中运行“notebook -setup”命令

出现如下提示：

```
>> notebook -setup
Welcome to the utility for setting up the MATLAB Notebook
for interfacing MATLAB to Microsoft Word
Choose your version of Microsoft Word:
[1] Microsoft Word 97
[2] Microsoft Word 2000
[3] Microsoft Word 2002 (XP)
[4] Exit, making no changes
Microsoft Word Version:
```

(3) 根据自己机器上所用 Word 的版本，在提示的冒号后填写数值，如“2”，并按回车键。

则出现以下提示，表示 Notebook 安装结束。

```
Notebook setup is complete.
```

如果安装程序不能找到所需的文件，则也会提示用户指定“winword.exe”和“normal.exe”文件的路径。

8.4.2 Notebook 的启动

1. 创建 M-book 文件

(1) 从 Word 中启动 Notebook

创建 M-book 文件的步骤如下：

选择 Word 窗口的菜单“文件”→“新建”，在出现的对话框中，选择“m-book.dot”图标，如图 8.3 所示的为 Word2000 的新建对话框，按“确定”按钮。当保存文件时，默认的文件名为“The MATLAB Notebook v1.doc”。

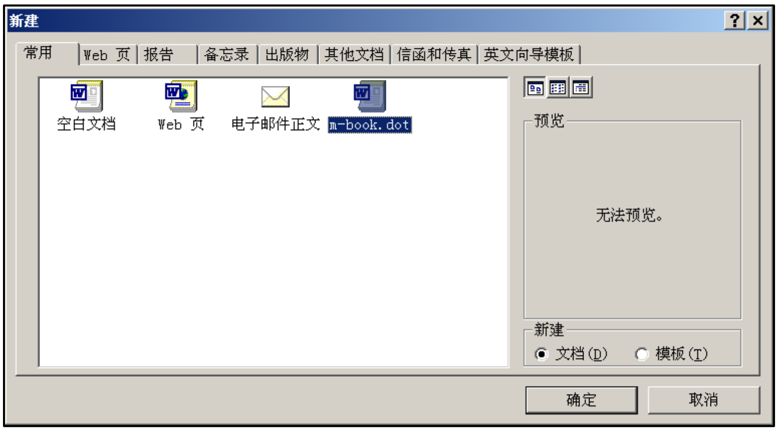


图 8.3 新建 word 文件对话框

(2) 从 MATLAB 中启动 Notebook

在 MATLAB 命令窗口输入“notebook”命令，就可以启动 Notebook。

语法：

notebook %打开一个新的 M-book 文档

notebook FileName %打开已存在的 M-book 文件

说明：FileName 应包括文件的完整路径和文件名。

例如，在命令窗口中打开已建立的 M-book 文件：

```
>> notebook 'C:\My Documents\MyMbook0801.doc'
```

2. M-book 的界面

M-book 的界面比普通的 Word 多一个“Notebook”菜单，Notebook 菜单项如图 8.4 所示。

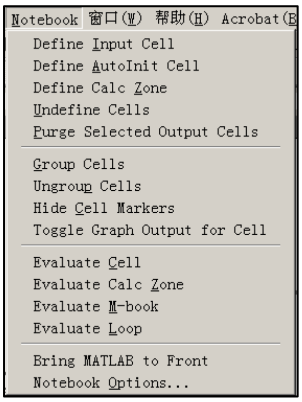


图 8.4 Notebook 菜单

其菜单项功能如表 8.1 所示。

表 8.1 Notebook 菜单项功能表

下拉菜单	组合键	功能
Define Input Cell	Alt+I	定义输入单元
Define AutoInit Cell	Alt+A	定义自动初始化单元
Define Calc Zone	Alt+Z	定义计算区
Undefine Cells	Alt+U	将单元转换为文本
Purge Output Cells	Alt+P	清除输出单元
Group Cells	Alt+G	定义单元组
Ungroup Cells	Alt+p	将单元组转换为单个单元
Hide/Show Cell Markers	Alt+C	隐藏/显示单元标志
Toggle Graph Output for cell		为每个单元锁定图形输出
Evaluate Cell	Ctrl+Enter	运行当前单元或单元组
Evaluate Calc Zone	Alt+Enter	运行当前计算区
Evaluate M-book	Alt+M	运行 M-book 中所有单元
Evaluate Loop	Alt+L	循环运行单元
Bring MATLAB to front		将 MATLAB 置于屏幕之前
Notebook Options…	Alt+O	定义输出显示选项

8.4.3 Notebook 的使用

Notebook 定义了几种格式来表示 MATLAB 的函数和命令，包括 AutoInit、Calc、Error、Input、Nograph、Output 以及 Word 默认的格式。Notebook 对常用各单元格式的默认设置见表 8.2 所示。

表 8.2 M-book 各单元样式

样式名	含义	字体
Input	输入单元	10 磅 (points)深绿色英文粗体 Courier New
Output	输出单元(数据和字符)	10 磅(points)蓝色英文细体 Courier New
AutoInit	自动初始化单元	10 磅(points)深蓝色英文粗体 Courier New
Error	出错提示	10 磅(points)深红色英文粗体 Courier New

1. 输入单元

(1) 只创建不运行输入单元

在英文状态下按普通的文本输入方式，输入 MATLAB 命令，可以是独立行或嵌在文本中，然后用光标选中，按组合键“Alt-D”，或选择菜单“Notebook”——“Define Input Cell”，则所选中的文本形式命令就变成了输入单元。

【例 8.9】在 M-book 文件中创建输入单元(用黑框表示 M-book 中的输入文本)。

在文本中输入以下文字：

利用：来生成向量，例如 x=1:0.1:3;

用光标选中“x=1:0.1:3;”文字，并按组合键“Alt-D”，则创建了输入单元显示如下：

利用：来生成向量，例如 **`x=1:0.1:3;`**

(2) 创建并同时运行输入单元

在英文状态下按普通的文本输入方式，输入 MATLAB 命令，然后用光标选中，按组合键“Ctrl-Enter”，或选择菜单“Notebook”——“Evaluate Cell”，则所选中的文本形式命令就会自动变成输入单元，并得出运算结果，即输出单元。

【例 8.9 续】创建并运行输入单元。

在文本中继续输入 MATLAB 命令：

y=sin(x)

用光标选中“y=sin(x)”文字，并按组合键“Ctrl-Enter”，则创建并运行输入单元，显示如下：

```
y=sin(x)
y =
Columns 1 through 7
0.8415    0.8912    0.9320    0.9636    0.9854    0.9975    0.9996
Columns 8 through 14
0.9917    0.9738    0.9463    0.9093    0.8632    0.8085    0.7457
Columns 15 through 21
0.6755    0.5985    0.5155    0.4274    0.3350    0.2392    0.1411
```

程序分析：其中的“x”的值来自 MATLAB 的工作空间，M-book 文件和 MATLAB 命令窗口共享一个工作空间；如果在命令“y=sin(x)”后加分号，则不以输出单元显示运行结果。

2. 自动初始化单元

创建自动初始化单元的方法是先将文本形式的 MATLAB 命令或已存在的输入单元用光标选中，然后选择菜单“Notebook”——“Define AutoInit Cell”，则选中的文本形式 MATLAB 命令就会自动变成 AutoInit 格式。

3. 单元组

创建单元组的方法：

(1) 将多行文本形式 MATLAB 命令用光标选中，然后选择菜单“Notebook”——“Define Input Cell”或“Notebook”——“Define AutoInit Cell”。

(2) 将多行独立的输入单元或自动初始化单元同时选中，然后选择菜单“Notebook”——“Group Cells”，就生成了第一个独立单元性质的单元组。

【例 8.9 续】创建单元组。

在 M-book 中输入以下文本：

```
利用：来生成向量，例如：x=1:0.1:3;
 y=sin(x)
画出正弦波形：
    plot(x,y)
```

选择所有的文本行，使用菜单“Notebook”——“Group Cells”创建单元组如下：

可以看到创建单元组后，中间“画出正弦波形”的文本内容移到单元组之后。

```
利用：来生成向量，例如：x=1:0.1:3;
y=sin(x)
plot(x,y)
画出正弦波形：
```

4. 输出单元

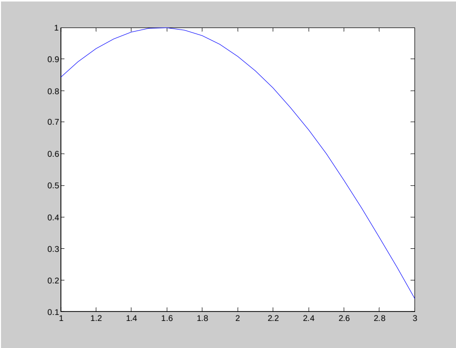
输出单元包含 MATLAB 的输出结果，包括数据、图形和出错信息。

【例 8.9 续】运行单元组，查看输出单元。

将单元组全部选中，按组合键“Ctrl-Enter”，就出现输出单元如下：

输出单元为 y 数值和图形，输出单元也用“[]”标志，输入单元组的运行结果也是输出单元组。

```
利用：来生成向量，例如：x=1:0.1:3;
y=sin(x)
plot(x,y)
    y =
Columns 1 through 7
0.8415    0.8912    0.9320    0.9636    0.9854    0.9975    0.9996
 Columns 8 through 14
 0.9917    0.9738    0.9463    0.9093    0.8632    0.8085    0.7457
Columns 15 through 21
0.6755    0.5985    0.5155    0.4274    0.3350    0.2392    0.1411
```



画出正弦波形：

5. 计算区

【例 8.10】使用计算区实现分支结构。

在 Word 文档中输入如下所有文本，选择菜单“Notebook”→“Define Calc Zone”，将所有文本定义为计算区，出现分节符；然后选定程序，再选择菜单“Notebook”→“Define Input Cell”，将程序定义为输入单元；光标在计算区中时，单击鼠标右键选择菜单“Notebook”→“Evaluate Calc Zone”，得出输出单元。

程序分析：在计算区中，输入单元和输出单元分别用“[]”包括，分节符在计算区的首尾。

1. If…else…end 结构

例：用 If…else…end 分支结构划分学生成绩为优、良、中、及格、不及格。

```
x=100*rand(1);
if x>=90
   score='优'
elseif x>=80
      score='良'
elseif x>=70
      score='中'
elseif x>=60
      score='及格'
else
      score='不及格'
end
score =
及格
```

6. 取消定义单元

取消定义单元的方法是，先选定单元，然后选择菜单“Notebook”——“Undefine Cells”，或当光标置于单元中时按“Alt-U”按钮，则该单元就变成“Normal”样式的文本。

8.4.4 Notebook 中 MATLAB 的使用

1. 工作内存的初始化

为了避免其它文件或命令窗口中变量的改变影响该文件，保证文件输入输出数据的一致性，可以用“clear”命令作为该文件的第一个自动初始化单元。

2. 整个 M-book 文件的运行

菜单“Notebook”——“Evaluate M-book”是运行整个 M-book 文件，即把文档中所有输入单元送到 MATLAB 中去运行。

3. 输出单元的格式控制

设置的方法为选择菜单“Notebook”——“Notebook Options…”，则出现如图 8.5 所示对话框。

(1) 输出数据的格式控制

在图 8.5 中的“Numeric Format”选项栏中，可以设置输出数据的格式，共有 8 种：“Short”、“Long”、“Hex”、“Bank”、“Plus”、“Shorte”、“Rational”。

(2) 输出数据间的空行控制

图 8.5 所示对话框中的“Loose”和“Compact”单选按钮是用来控制输入单元与输出单元之间的空白区间。

(3) 图形的嵌入控制

“Embed Figures in M-book”复选框是用来决定输入单元中的绘图命令是否向 M-book 文档输出图形。

(4) 嵌入图形的大小控制

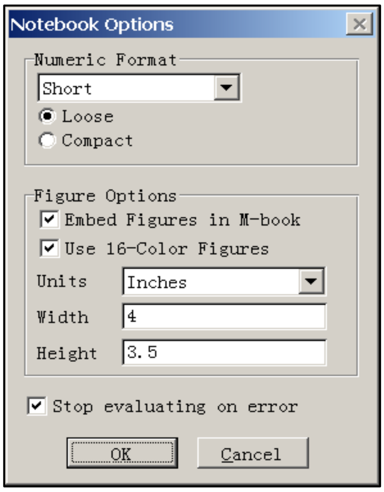


图 8.5 Notebook Options 对话框

通过对图 8.5 的“Figure Options”框中的三个栏目“Units”、“Width”、“Height”的设置，可以控制嵌入图形的大小。“Units”有三个选项：Inches、Centimeters 和 Points。

(5) 嵌入图形的打印输出控制

(6) 嵌入图形的背景色控制

在默认情况下，正常嵌入 Word 的图形背景色应是“灰/白”的。

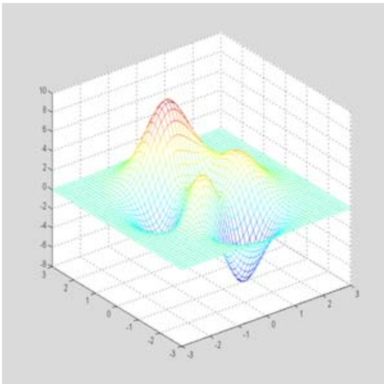
如果所嵌入的图形出现“灰/黑”背景色，可采取以下两种措施：

- 打开“Notebook Options”对话框，将“Embed Figures in M-book”复选框置于“选中”状态，单击“OK”按钮，然后重新运行输入单元。
- 在 MATLAB 命令窗口中，运行以下命令：

【例 8.11】绘制三维 peaks 函数图形。

选择菜单“Notebook”——“Notebook Options…”，将“Width”和“Height”都设置为 2，输入如下文本，并用“Evaluate M-book”命令运行该文本，则在 M-book 中显示如下图形：

绘制 peaks 函数的三维曲面图`[x,y,z]=peaks;`
`mesh(x,y,z)`



4. 单元的循环运行

Notebook 提供了循环运行单元的命令，先选定需要循环运行的输入单元，然后选择菜单“Notebook”——“Evaluate Loop”，就会出现如图 8.6 所示的对话框。

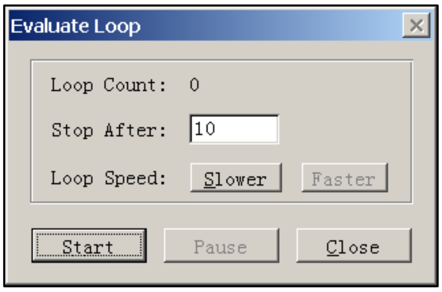


图 8.6 循环运行对话框

5. 删除 M-book 文件所有选中的输出单元

当在撰写报告或布置作业时，如果需要删除所有的输出单元，可以选择菜单“Notebook”——“Purge Selected output Cells”，则会删除选中的所有输出单元。